Assessing the Reproducibility of Clustering of Molecular Dynamics Conformations on Self-Organizing Maps

By

Michelle A. Eisner

A THESIS SUBMITED IN PARTIAL FULFILMENT OF

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In

The Faculty of Mathematics and Sciences

Department of Chemistry

BROCK UNIVERSITY

October 21, 2015

Abstract

The goal of most clustering algorithms is to find the optimal number of clusters (i.e. fewest number of clusters). However, analysis of molecular conformations of biological macromolecules obtained from computer simulations may benefit from a larger array of clusters. The Self-Organizing Map (SOM) clustering method has the advantage of generating large numbers of clusters, but often gives ambiguous results. In this work, SOMs have been shown to be reproducible when the same conformational dataset is independently clustered multiple times (~100), with the help of the Cramérs V-index (C_v). The ability of C_v to determine which SOMs are reproduced is generalizable across different SOM source codes. The conformational ensembles produced from MD (molecular dynamics) and REMD (replica exchange molecular dynamics) simulations of the penta-peptide Met-enkephalin (MET) and the 34 amino acid protein human Parathyroid Hormone (hPTH) were used to evaluate SOM reproducibility. The training length for the SOM has a huge impact on the reproducibility.

Analysis of MET conformational data definitively determined that toroidal SOMs cluster data better than bordered maps due to the fact that toroidal maps do not have an edge effect. For the source code from MATLAB, it was determined that the learning rate function should be LINEAR with an initial learning rate factor of 0.05 and the SOM should be trained by a sequential algorithm. The trained SOMs can be used as a supervised classification for another dataset.

The toroidal 10×10 hexagonal SOMs produced from the MATLAB program for hPTH conformational data produced three sets of reproducible clusters (27%, 15%, and 13% of 100 independent runs) which find similar partitionings to those of smaller 6×6 SOMs. The χ^2 values produced as part of the C_v calculation were used to locate clusters with identical conformational memberships on independently trained SOMs, even those with different dimensions. The χ^2 values could relate the different SOM partitionings to each other.

Acknowledgements

I would like to thank Dr. Heather Gordon for her all her guidance and support over the last few years. I would also like to thank my committee members Dr. Art van der Est and Dr. Hongbin (Tony) Yan for their input and suggestions during my research. I would also like to thank the members of Dr. Gordon's lab for their support. Completing my MSc has me reminding of a saying my Bubbie (Grandmother) told be her Principal wrote in her year book long ago. "When you laugh the world laughs with you, when you cry you cry alone." True friends and family get you through the hard times. I would like to whole heartily thank Brad and Egor for their help and support over the years, and Alex for listening and believing in me, as well as my family for their support. This work was made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET: www.sharcnet.ca) and Compute/Calcul Canada.

Table of Contents

List of Tables

List of Illustrations

List of Abbreviations

Abbreviation	Full Title
SOM	Self-Organizing Map
IUP	Intrinsically Unstructured Proteins
INSSQ	Within-group-sum-of-squares Objective Function
BTWSSQ	Between-group-sum-of-squares Objective Function
NEIGH	Neighbourhood Objective Function
TE	Topological Error
QE	Quantization Error
maxmin	Maximum/Minimum Initialization
rand	Random Initialization
rvec	Random Vector Initialization
seq	Sequential Training Algorithm
batch	Batch Training Algorithm
rect	Rectangular Geometry
hexa	Hexagonal Geometry
CHARMM	Chemistry at Harvard Macromolecular Mechanics
NAMD	Nanoscale Molecular Dynamics
MD	Molecular Dynamics
REMD	Replica Exchange Molecular Dynamics
PBC	Periodic Boundary Conditions
PME	Partial Mesh Ewald
C_v	Cramérs V-index
χ^2	Chi squared
SI	Similarity Index
rSI	Revised Similarity Index
BMU	Best Matching Unit
SBMU	Second Best Matching Unit
PC	Principal Component
PCA	Principal Component Analysis
dPCA	dihedral Principal Component Analysis
MET	Met-enkephalin
hPTH	Human parathyroid hormone
NCBD	Nuclear Co-activator Binding Domain

1 Introduction

1.1 Clustering

The goal of most clustering algorithms is to find the optimal (i.e. fewest) number of clusters into which the dataset can be partitioned.[1] There are two types of clustering: supervised and un-supervised.[1] Supervised clustering is the process of grouping data, where the type of groups is predetermined. This means the data are previously labeled (e.g. types of dogs) or a predefined set of characteristics are used to divide the dataset into groups.[1] Un-supervised clustering involves dividing the data into groups without knowing what the groups should be. Un-supervised clustering finds hidden structures in the data. The clustering algorithm discussed in this work, self-organizing maps (SOM), is an un-supervised partitional clustering method.[1–4]

The two types of un-supervised clustering algorithms are partitional and hierarchical.[1] Hierarchical clustering usually starts with each data vector in its own cluster and then merges the clusters based on a distance criterion.[5] As the iterative process continues, the groups of data vectors eventually merge until one super-cluster is generated. The merging of data vectors is usually depicted as a tree or dendrogram. It is up to the user to interpret the dendrogram and decide on the optimal number of clusters.[1]

A partitional algorithm breaks the data into a predetermined number of groups and finds the optimal cluster membership for each data vector.[5] The partitional algorithm usually uses an iterative process. The process starts by randomly assigning data to the predefined clusters. Once all the data are assigned, the quality of the clusters is evaluated by cluster criteria. Then the data are reassigned based on similarity to the clusters. The reassignment of data and updating of clusters continues until the data stop being reassigned to different clusters (i.e. convergence is

reached). A partitional clustering algorithm does not always generate the same partitioning of a single dataset each time it is run. Therefore, it is up to the user to run the program several (many) times and then to interpret which run produced the optimal clustering partition.

1.2 Evaluating Clustering by Objective Functions

An objective function is a numerical quantity used to assess the quality of the clustering that is produced by an algorithm. Normally, there are two main goals when designing an objective function:[6] (1) to assess the compactness of the clusters, and (2) to assess the distance between clusters. If an algorithm produces good clusters, an objective function for cluster compactness should result in a small quantity, while the distance between clusters should be a large quantity.

Two objective functions used in this work are the within-group-sum-of-squares (INSSQ) and between-group-sum-of-squares (BTWSSQ). The INSSQ assesses the compactness of clusters by computing the sum of mean squared distances between all the data vectors within each cluster: [1,6]

$$\underset{(INSSQ)}{Objective\ Function} = \sum_{k=1}^{\eta^2} \frac{\sum_{i=1}^{S_k-1}\sum_{j=i+1}^{S_k}(V_j - V_i)^2}{S_k(S_k-1)/2} \qquad \textbf{[1]}$$

where V_i and V_j are two different data vectors that are members of cluster k, η^2 is the total number of clusters, S_k is the total number of vectors in cluster k. The BTWSSQ assesses the sum of squared distances between clusters by computing the squared distances between data vectors assigned to different clusters: [1,6]

$$\underset{(BTWSSQ)}{Objective\ Function} = \sum_{m=1}^{\eta^2-1} \sum_{k=m+1}^{\eta^2} \sum_{i=1}^{S_m} \sum_{j=i}^{S_k} (V_j - V_i)^2 \qquad \textbf{[2]}$$

where S_m is the total number of vectors in cluster m, the vector V_i belongs to cluster m while the vector V_j belongs to cluster k.

1.3 Protein Conformations

Proteins are complex molecules that are polymers of amino acids. The amino acids are held together by peptide bonds. Determining the three-dimensional structure of proteins gives insight into protein function and/or interactions with other macromolecules or ligands.[7,8] The three-dimensional structure of a protein is called a conformation. Conformations can vary in a protein by different folding patterns of the backbone.[9] Figure 1 shows a section of a protein. The black circles, excluding the one labeled "R", are the atoms making up the backbone ((O=C)= carbonyl, N=nitrogen, C_α=α-carbon). The black circle "R" represents the amino acid sidechains, which differ for the 21 naturally-occurring amino acids. The backbone can be described by rotatable angles (dihedral angles) or atom positions, which explain the main conformational changes of a protein.[10] There are three dihedral angles in the protein backbone: φ, ψ, and ω (Figure 1). A dihedral angle is defined by four sequentially bonded atoms (φ (C-N-C_α-C), ψ (N-C_α-C-N), and ω (C_α-C-N-C_α)). The dihedral angle ω is not used in clustering to describe the backbone because the peptide bond is planar and is only found to be close to 0° (*cis*) or 180° (*trans*).[11]

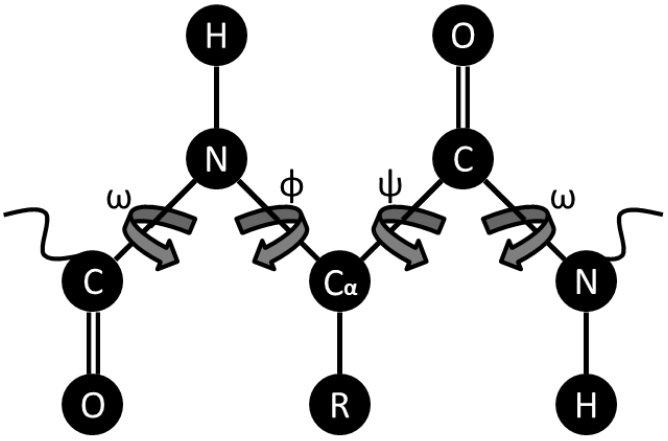


Figure 1: Representation of the protein backbone showing the ϕ, ψ, and ω dihedral angles.

The angles φ and ψ for a given amino acid are inter-dependent. Ramachandran determined this dependence.[12] The dependency of dihedral angles can be depicted by a Ramachandran plot (Figure 2), where φ is the horizontal axis and ψ is the vertical axis. The Ramachandran plot can show protein secondary structure. There are two main types of secondary structure: α-helixes and β-sheets. These structures are formed when certain φ and ψ angles are paired. The α-helixes are in the area of the plot at (φ,ψ) ~(-57°, -47°) and β-sheets are in the area (φ,ψ) ~(-119°, 113°).[13] The protein tertiary structure is the three-dimensional structure of a single protein chain and quaternary structure is the arrangement of a protein complex (i.e. different subunits).

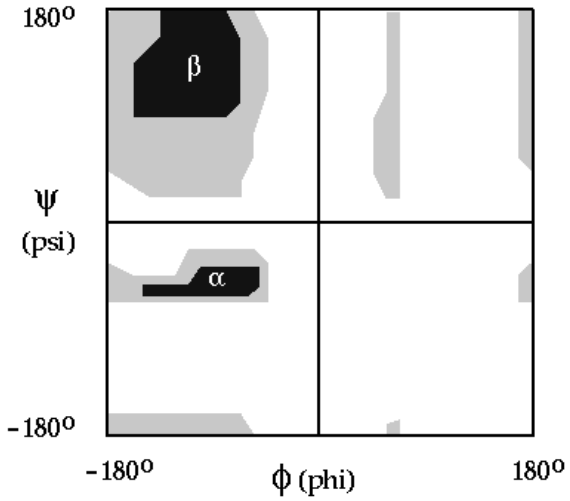


Figure 2: The Ramachandran plot of allowed and disallowed regions. The black regions are (ϕ,ψ) values which are normally found, while the grey regions correspond to the outer limit.[14] White regions are "disallowed".

1.3.1 Protein Conformational Distributions

In solution proteins are not stationary, they form many different conformations; the collection of all conformations of a protein is a conformational distribution. The more secondary, tertiary, and/or quaternary structure(s) hold the protein together, the narrower the conformational distribution (less variation in the structure). Some proteins, like the ones examined in this work, have a broad conformational distribution and little to no secondary, tertiary, or quaternary structure(s). These proteins are referred to as intrinsically unstructured proteins (IUP). A single conformation cannot describe the breadth of the conformational distribution of an IUP.

The Boltzmann distribution of states is derived by statistical mechanics to describe the probability that the system will be in a certain state.[15] The Boltzmann distribution of states for a homogenous single particle fluid is:[15]

$$f(\vec{R}_i) = \frac{\exp\{-U(\vec{R}_i)/(k_B T)\}}{\sum_{i=1}^{M} \exp\{-U(\vec{R}_i)/(k_B T)\}}$$

where $\vec{R}_i = \vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N$, is the i^{th} conformation of N particles. Integration over all Mallowed states gives:

$$f(\vec{R}_i) = \frac{N}{V}\left(\frac{1}{2\pi m k_B T}\right)^{3/2} \exp\{-U(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)_i/(k_B T)\} \quad \textbf{[3]}$$

where $f(\vec{R}_i)$ is the frequency with which the i^{th} conformation or state appears within the distribution, U is the potential energy and is dependent on the positions of N particles, k_B is the Boltzmann constant, and T is the temperature of the system in Kelvin. The term $\frac{N}{V}\left(\frac{1}{2\pi m k_B T}\right)^{3/2}$ is a normalization factor, M is the number of states, N is the number of particles, V is the volume, and m is the mass of a single particle in the fluid. The expression for the Boltzmann distribution of states for a protein is more complicated than equation [3] due to the many different types of atoms and their interconnections.

Thermodynamic properties (e.g. heat capacity) and time-dependent properties (e.g. diffusion coefficient) can be determined by both experiment and simulation. However, in order to determine these properties by simulation, the simulation must sample the Boltzmann distribution of states.

Molecular Dynamics (MD) simulations simulate the positions of molecules over a period of time. MD simulations are a way to explore the conformational diversity accessible to models of proteins and other biomolecules.[16] MD simulations produce a trajectory file that contains atomic coordinates at discrete simulation time intervals.[17] MD simulations assume that when a simulation is run long enough, the virtual model of the system will explore all accessible conformational space within the energy barriers and this will follow the Boltzmann distribution

of states. Properties like heat capacity can be found by computing an average across the simulation:[18]

$$C_P = \frac{< U^2 > - < U >^2}{k_B T^2} \quad \textbf{[4]}$$

where U is the potential energy and is dependent on the atom positions, k_B is the Boltzmann constant, T is the temperature of the system in Kelvin,< > is a representation for average over the conformational samples collected throughout the simulation. The experimentally determined heat capacity (by calorimetry) and the simulation determined heat capacity can be compared, to determine the accuracy of the simulation. The simulation can also predict values for thermodynamic properties for systems for which no experimental measurements are available or where the experimental data is difficult or impossible to obtain.[19]

1.4 Clustering of Molecular Conformations Obtained from Molecular Dynamics (MD) Simulations

Biological macromolecules have been simulated to gain insight into the conformations assumed *in vivo* as well as their thermodynamic properties. Advances in technology have enabled simulation time to increase to a nanosecond time scale. The increase in simulation time produces a larger number of conformations that need to be analyzed, to gain insight into the function and structure of the macromolecule being simulated. One way the analysis can be done is grouping (clustering) like conformations together. In theory, conformations that have similar structure occupy the same free energy well; clustering the conformations based on their geometry can take a larger dataset and narrow it down to a few main conformations.

Our eyes are good at finding patterns. However they can only perceive them in three- or two-dimensional space. Proteins are high-dimensional data, where each dimension is a descriptor

usually of the geometry (dihedral angles). High-dimensional data is anything over three-dimensions. Since our eyes cannot easily sort data with more than three-dimensions, computer programs can be used to cluster high-dimensional data, where the protein conformational ensemble is the dataset, and the individual conformations are described as data vectors. Each dimension in the vector is a descriptor. Clustering reduces the dimensionality of the data by generating groups of like data, so that data is more alike in the same group than any other group, and one conformation can describe the entire group.

To find functionally relevant conformations or smaller subsets of like conformations in the trajectory file a clustering algorithm can be applied. The majority of clustering algorithms cluster the conformations based on some description of the molecular geometry (e.g. Cartesian coordinates of atoms or dihedral angles). MD conformational ensembles can be thought of as a dataset of vectors with n-dimensions, where the dimensions describe the molecular geometry, and the number of vectors in the dataset is the number of conformations. Some of the clustering algorithms used to cluster MD trajectories are hierarchical,[5] k-means,[20] fuzzy clustering,[2] and SOMs.[21] The different algorithms have different criteria for clustering the dataset; no one algorithm is considered better than any other for all datasets.

The hierarchical method is an un-supervised clustering technique that unites data in a step wise manner; once data points are united within a group they cannot be separated. These groups continue to merge until k clusters are reached. This method does not produce a cluster centre. To determine a representative geometry of the conformations within a cluster, a criteria for picking one of the member conformations must be used. Shenkin and McDonald used hierarchical clustering to test their method of locating clusters using two molecular dynamics simulations of pentane.[5] Both simulations produced 200 structures. They were trying to cover all

conformational space and used clusters of dihedral angles to determine if they did. The possible dihedral angles are 180° (*trans*), -60°(*gauche-*), 60° (*gauche+*); these angles are combined to give the four unique possibilities (*trans*:*trans*, *gauche+*:*gauche-*, *gauche(+/-)*:*gauche(+/-)*, *trans*:*gauche(+/-)*). They only found three out of the four minima for the (C_1-C_2-C_3-C_4) and (C_2-C_3-C_4- C_5) dihedral angles for n-pentane. They determined the simulation was not run for a sufficient length of simulation time since *gauche+*:*gauche-* was never located. [5]

The k-means algorithm is an un-supervised clustering technique that partitions the data into k clusters.[20] The data belong to the cluster with the nearest mean value. K-means clustering requires an initialization of the cluster centres; most algorithms randomly select values for these centres. The data are assigned to the closest cluster centre (usually by computing Euclidean distance of data vector to cluster centre vector). The cluster centres are updated based on the current cluster membership (i.e. mean of data vectors). The two steps of data reassignment and cluster centre updating are repeated until convergence is reached. One of the disadvantages is that the clusters found are not always the optimal partitioning of the data. An objective function is used to help find optimal clusters. Kasson *et al.* used k-means to determine the different conformational states used to fuse two 14nm vesicles of 1-palmitoyl 2-oleoyl phosphatidylethanolamine (POPE) into one large vesicle. [22] The conformations of the POPE vesicle(s) were generated by ten independent coarse-grained MD simulations. More than 85000 structures were produced. Eight clusters were located using k-means clustering to help describe the vesicle fusion states.[22]

Fuzzy clustering attempts to describe overlapping groups where data vectors can belong to more than one group. Gordon and Somorjai used fuzzy clustering to cluster conformations of parathyroid hormone produced from MD simulations.[2] Eight simulations were run and ~1000

conformations were taken from each, to be clustered by their inter-alpha-carbon distances. One to three clusters were determined for each simulation.[2] The authors determined that the independent simulations did not explore the same regions of conformational space.

The sharing of data by different clusters described by fuzzy clustering is similar to the training of a self-organizing map (SOM), another partitional clustering method. SOMs organize the clusters based on topology. This means that clusters closer to each other on the two-dimensional map comprising the SOM are more similar to each other than groups farther away.

2 Self-Organizing Maps

Self-Organizing Maps (SOMs) can be used to classify the data into clusters. SOMs, also referred to as Kohonen Neural Networks (KNN), were first described by Teuvo Kohonen and are a way of classifying high-dimensional data on a low- or two-dimensional map.[23] This method uses two approaches: clustering and projection. Clustering refers to putting like data into the same classification; projection refers to the ability to visualize the results on a low- or two-dimensional map.[24] The classification of the dataset is done by a computer algorithm, which simulates un-supervised learning through competition (Figure 3).[25] The mapsize of an SOM designates the number of clusters (or nodes) in the map (e.g. a 5×5 map has 25 nodes). A square mapsize is $\eta \times \eta$ and contains η^2 nodes. Each node has a nodal centre which is a representation of what data have been assigned to the node. The nodal centres possess the same dimensionality as the training set. The initialization of the training process can be done in a variety of ways (Chapter 2.1). The data vectors (training set) are compared to the node centres. Competitive learning is a process to determine which node "wins" each data vector. The nodal centre which is closest (most similar) to each data vector is assigned that vector to its nodal membership. The

node to which each data vector belongs is called the Best Matching Unit (BMU).[23] The learning or training process refers to the iterative process of partitioning the data among the η^2 nodes of the SOM (Figure 3). The neighbourhood of a node is defined as the topologically adjacent nodes whose contents influence the value of the nodal centre. The neighbourhood of each node is dependent on the iteration, the lower the iteration step, the larger the neighbourhood. For example, initially the entire map might constitute the neighbourhood of each node. At the end of the training process, only the immediately adjacent nodes might be the neighbourhood. As the training progresses, the nodes gradually become more sensitive to the input categories, because the data vectors contribute to the updating of fewer nodal centres.

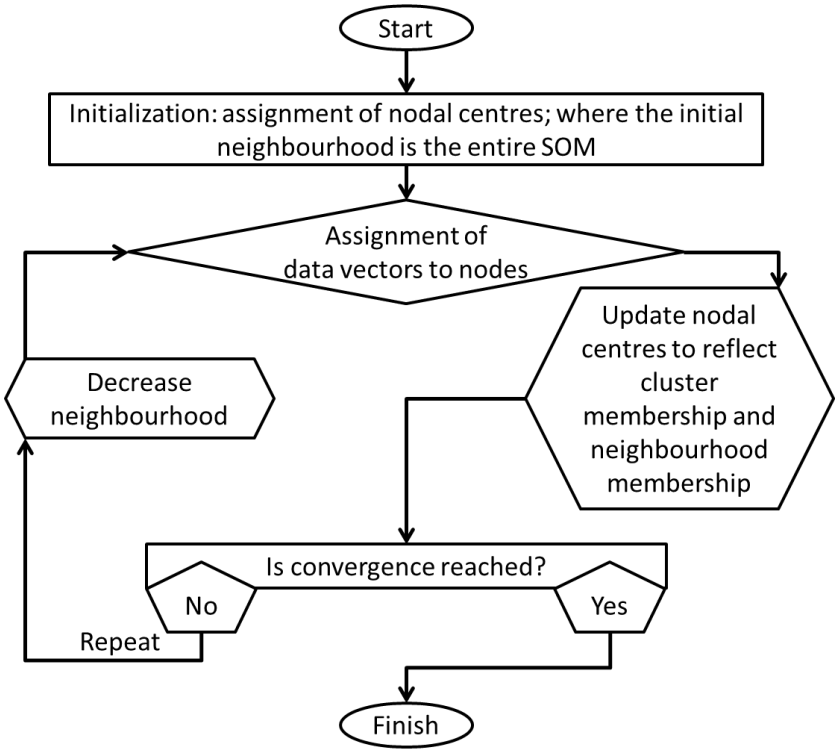


Figure 3: Flowchart of the SOM learning process.

SOMs can also be used as a supervised clustering method. This work implements this novel application for an SOM to be used as labels for supervised clustering to compare conformational distributions obtained by MD simulations. This method can be used when

comparing different datasets of the same system to determine quantitatively how similar the datasets are. Figure 4 shows a flow chart of how a previously trained SOM can be used to classify new data. The un-supervised training is the same as discussed in Figure 3. At the end of this training two vectors are produced for each node: a nodal centre vector and a nodal tolerance vector. The nodal centre is a weighted average of all the data vectors within that node's membership and the members of its neighbourhood. The nodal tolerance is the (Euclidean) distance between a single data vector in the nodal membership which is farthest away from the nodal centre. These vectors are used as criteria for the supervised training by the map of a different dataset. The new data vectors are placed into the nodal membership of the closest nodal centre by Euclidean distance. However, they are only accepted into the nodal membership if they are closer to the nodal centre than the nodal tolerance. If the distance of a data vector is greater than the nodal tolerance, the new data vector is not accepted onto the map. That is, that the data vector is not well-described by the dataset used to train the SOM. This means that only a percentage of the new dataset can be described by the SOM classifiers. If the dataset used to train the SOM is plotted on the SOM, 100% of the dataset should be accepted into the nodal memebrships corresponding to their original placement.

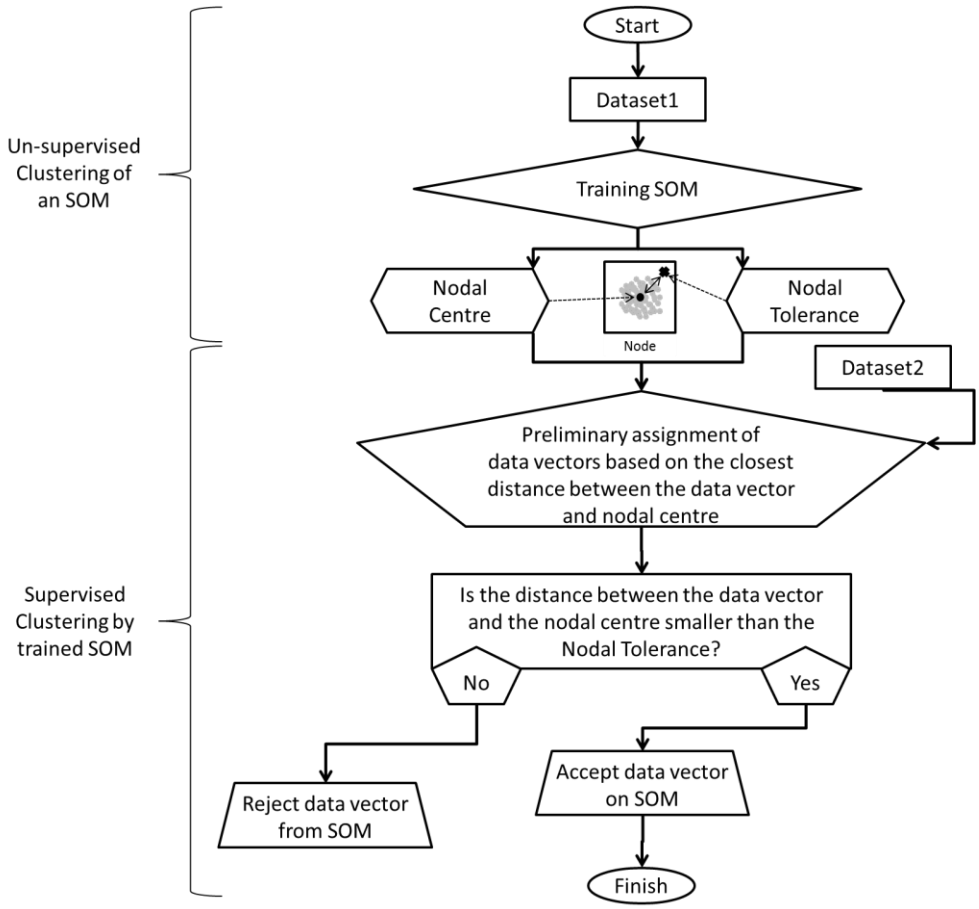


Figure 4: Flowchart of supervised clustering using an SOM.

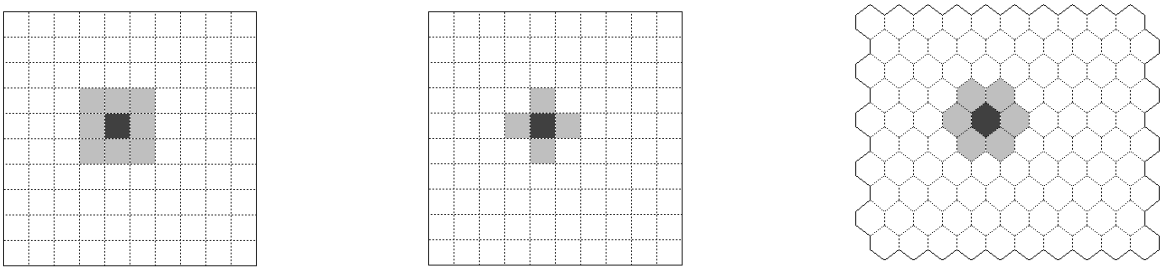
2.1 Nodal Geometries

The SOM can have many types of geometry. Figure 5 illustrates 10×10 SOMs with rectangular and hexagonal derived nodes.[26] The most commonly used SOMs have two-dimensional nodes. López and Ramos found that there are other geometries which are better at clustering by generating groups that represent the input dataset more closely.[26] However, the rectangular and hexagonal geometries are the easiest to program within a computer algorithm. The nodal geometry in the two-dimensional map dictates the nature of the topological neighbourhood. The contents of nodes in the current neighbourhood of each node are used to update the value of its nodal centre. The neighbourhood of each node is defined as a subset of other nodes of the map between which the data contents can be exchanged during the training

process. Figure 5 shows a best matching unit (BMU) and its neighbours in the smallest neighbourhood of two rectangular maps and one hexagonal map. The rectangular SOM can have four or eight nodes in the smallest neighbourhood of the BMU, depending on how the neighbourhood radius function is defined.

In Rectangular (A), the neighbourhood is defined by the indexed proximity of other nodes to the BMU. That is, if the BMU has a location index (i, j) then its smallest neighbourhood includes nodes$(i-1, j+1)$, $(i, j+1)$, $(i+1, j+1)$, $(i-1, j)$, $(i+1, j)$, $(i-1, j-1)$, $(i, j-1)$, and $(i+1, j-1)$. In Rectangular (B), the Euclidean distance of the nodes $(i-1, j+1)$, $(i+1, j+1)$, $(i-1, j-1)$, and $(i+1, j-1)$ to the BMU is larger than that of $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$.

It is thought that since each hexagonal node in a SOM has six equivalently distanced neighbours, they cluster data better than rectangular SOMs. However, this has not been proven.[27]



Rectangular (A) **Rectangular (B)** **Hexagonal**

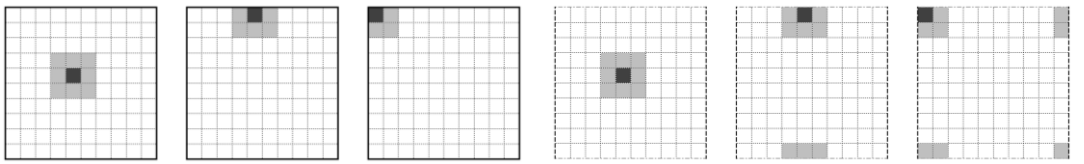
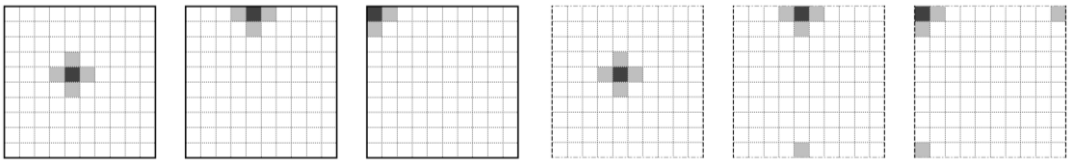
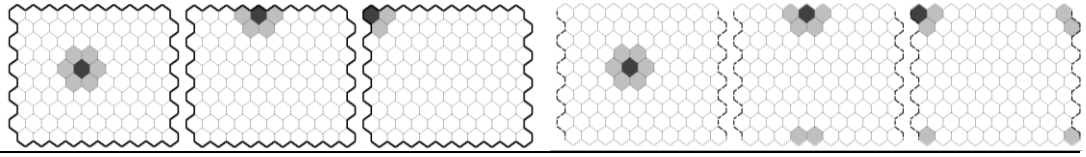
Figure 5: The difference between the nodal geometries of a 10×10 SOM. The dark grey box is the BMU and the light grey boxes are the nodes in the smallest neighbourhood of the BMU. Two different neighbourhood radius functions are shown for rectangular nodes: the number of neighbours for Rectangular (A) is eight, while for Rectangular (B) is four. The Hexagonal BMU has six nodal neighbours.

2.2 SOM Boundaries

Two different boundaries used on the SOMs are bordered and toroidal. The difference between these map boundaries is best appreciated when the BMU is on the edge or corner of the

SOM. Toroidal maps connect the map edges: the top edge of the SOM is connected to the bottom edge of the SOM and the right edge of the SOM is connected to the left edge of the SOM (Table 1). Table 1 shows the smallest neighbourhood around a BMU. When the BMU is in the centre of the SOM, the smallest neighbourhood is the same for both boundary conditions. However, when the BMU is on the edge or on the corner of the SOM, the boundaries treat the neighbourhoods differently. In Table 1, the toroidal map connections are demonstrated with shade and line variations. The edges with the same shades and lines are considered connected. The bordered map has fewer nodes surrounding the edge or corner BMU resulting in an "edge effect".[4] The toroidal map has the same number of nodes in the neighbourhood of the BMU regardless of the BMU location. On bordered SOMs, there is a tendency for data vectors to pile up in the edge nodes; this influences the value of the nodal centre.

Table 1: The effect of different boundaries of a 10×10 SOM on the smallest nodal neighbourhood. The dark grey box is the BMU and the light grey boxes are the nodes in the smallest neighbourhood of the BMU. These images are of three different conditions. The BMU is located in three locations: interior (left), edge (middle), and corner (right) in each of the two panels.

Nodal Geometry	Bordered	Toroidal
Rectangular (A)		
Rectangular (B)		
Hexagonal		

The SOM algorithm uses distance to compute the contribution of each node to the nodal centre of its surrounding nodes. One method computes the Euclidian distance between the indexed locations of the nodes. This distance is defined differently for the bordered and toroidal SOMs.[28] Equation [5] defines $dist_B$ as the distance calculated on a bordered map and $dist_T$, equation [6], is the distance calculated on a toroidal map:

$$dist_B = \sqrt{\|x - xx\|^2 + \|y - yy\|^2} \quad \textbf{[5]}$$

$$dist_T = \sqrt{\left(\|x - xx\| - \eta \cdot int\left(\frac{\|x - xx\|}{\eta} + 0.5\right)\right)^2 + \left(\|y - yy\| - \eta \cdot int\left(\frac{\|y - yy\|}{\eta} + 0.5\right)\right)^2} \quad \textbf{[6]}$$

where (x,y) are the nodal indices on the two-dimensional SOM, (xx, yy) are the nodal indices of the BMU. The notation $int(\quad)$ refers to integer division; the addition of 0.5 eliminates rounding problems and $\|\quad\|$ is the absolute value. For example, in Table 1, the toroidal SOM with a BMU on the edge (the 1st and 2nd rows for column 5) has neighbour(s) on opposite sides of the map. This is because instead of being nine nodal indices away in the y direction, they are only one nodal index away. For this example the BMU has an index of $xx = 5$ and $yy = 1$, equation [6], and it is compared to the node $x = 5$ and $y = 10$. These nodes are in the smallest neighbourhood to each other, since the distance between the nodes is one.

$$dist_T = \sqrt{0 + \left(\|10 - 1\| - 10 \cdot int\left(\frac{\|10 - 1\|}{10} + 0.5 \right) \right)^2}$$

$$dist_T = \sqrt{\left(9 - 10 \cdot int(1.4)\right)^2}$$

$$dist_T = \sqrt{(9 - 10 \cdot 1)^2}$$

$$dist_T = 1$$

2.3 Map Initialization

Ideally, the original assignment or initialization of an SOM should not have an impact on the final trained maps.[29] In the work described here, three different map initializations have been tested: maximum/minimum, random assignment, and random vector assignment. These initializations are described in the following sections.

2.3.1 Maximum/Minimum

The map initialization by Maximum/Minimum (maxmin) generates two vectors reperesenting the limits of the n-dimensional vectors of the dataset. One vector contains the minimum values for each of the parameters, while the other vector contains the maximum values. Each node of the SOM is assigned an initial value for its nodal centre randomly chosen between the maximum and minimum values for the n-dimensional parameters. The updating of the nodal centres and the re-assignment of data membership to nodes follows the subsequent SOM algorithms for unsupervised learning by competition.

The maxmin initialization can generate initial nodal centre values that do not reflect a typical member of the dataset because each initial parameter value is generated independently from the rest. This method does not reflect correlations among the n-dimensional parameters of

the dataset. During the training process a possible consequence is that a nodal centre may never be assigned any members since it is not close enough to any members of the dataset, eliminating a potential cluster and reducing the total number of clusters.

2.3.2 Random Assignment

The initialization by random assignment (rand) assigns each member of the dataset to randomly chosen nodes of the SOM. The average values for each parameter computed over the values of the randomly selected nodal memberships become the nodal centres. The updating and the assignment of vectors to nodes follows the SOM algorithms for unsupervised learning by competition.

Initialization by random assignment can also generate initial nodal centre values that do not reflect a typical member of the dataset, due to correlation among parameters of the dataset. This method has the same consequences as maxmin, that is, at the end of the training period, one or more nodes may have zero members.

2.3.3 Random Vector Assignment

The initialization by random vector assignment (rvec) assigns random members from the dataset to be the initial values of the nodal centres. This initialization is the only one that generates initial nodal centres that do reflect a typical member of the dataset.

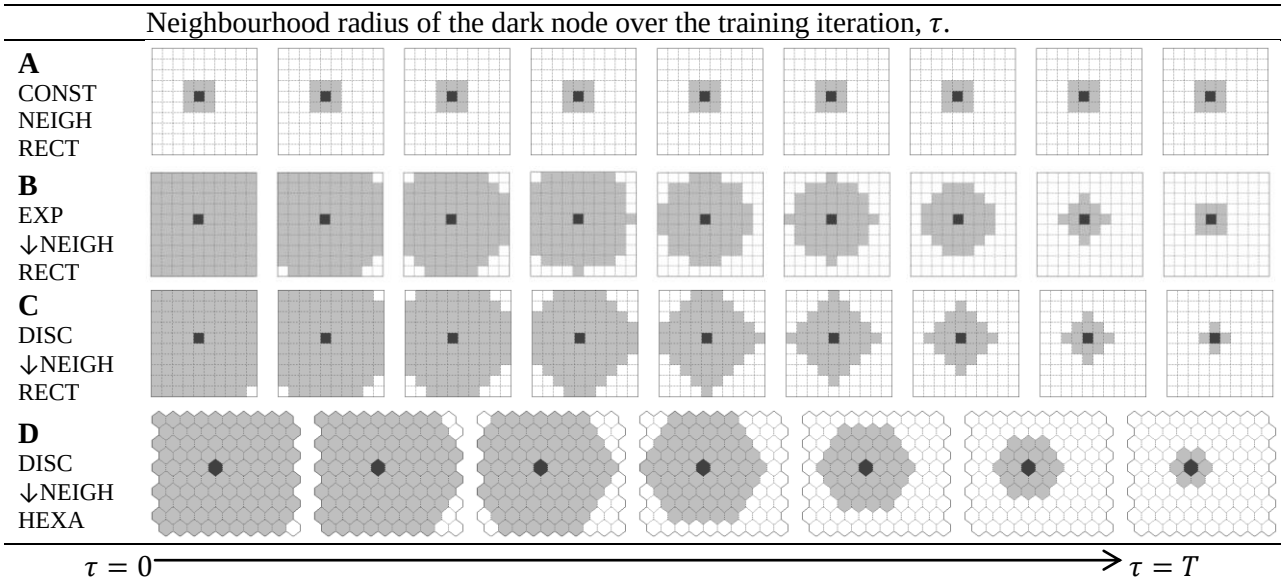
2.4 Map Training Algorithms

Most SOMs employ equation [7] as the neighbourhood function that determines which nodes are in the BMU's neighbourhood. The neighbourhood function determines the weight with which a data vector contributes to the new nodal centre:

$$h_{ci}(\tau) = \exp\left(\frac{-dist^2}{2\eta_\tau{}^2}\right) \quad \textbf{[7]}$$

where $dist$ is the distance between the BMU, c, and another nodal centre, i, in the neighbourhood, and $\eta_\tau$is the neighbourhood radius function at iteration τ (described later in this section). The range of τ is between 1 and T, where T is the maximum training length. Data vectors contribute more to their BMU (i.e. the node to which they are currently assigned) than to other nodes on the SOM. Data vectors outside the neighbourhood of the BMU will have zero contribution to the value of the nodal centre. This means data vectors closer to the node of interest will contribute more than vectors further away. During SOM training, the extent of the nodal neighbourhood is often decreased (Figure 4). As the number of iterations increase, the size of the nodal neighbourhood decreases/contracts. It is common for the initial neighbourhood of each node to encompass the entire map. Table 2 depicts the change in the nodal neighbourhood of a central node of a 10×10 rectangular and hexagonal SOM for different neighbourhood contraction schemes.

Table 2: A pictorial representation of the different neighbourhood radius functions over the iterations of a 10×10 SOM. The dark grey box is the BMU and the light grey boxes are the nodes in the neighbourhood of the BMU. [A] Constant neighbourhood rectangular SOM (CONST NEIGH RECT), [B] Exponential decreasing neighbourhood on a rectangular SOM (EXP ↓NEIGH RECT), [C] Discrete decreasing neighbourhood on a rectangular SOM (DISC ↓NEIGH RECT), and [D] Discrete decreasing neighbourhood on a hexagonal SOM (DISC ↓NEIGH HEXA)



Three neighbourhood radius functions, η_τ, are used in Table 2. The neighbourhood radius function depicted in Table 2A is

$$\eta_\tau = 1 \quad \textbf{[8]}$$

for a constant nodal neighbourhood (CONST NEIGH). For Table 2B,

$$\eta_\tau = \eta \cdot \exp\left(\frac{-\tau}{(T/\ln(\eta))}\right) \quad \textbf{[9]}$$

uses an inverse exponential function (EXP↓NEIGH). Table 2C and Table 2D use

$$\eta_\tau = \eta + (\tau - 1) \cdot \left(\frac{1-\eta}{T-1}\right) \quad \textbf{[10]}$$

for a contracting discrete neighbourhood (DISC↓NEIGH). In equations [8]-[10], η is one dimension of the map, τ is the iteration number and T is the training length.

The neighbourhood radius function, η_τ, is used in defining the current neighbourhood function, $h_{ci}(\tau)$, equation [7]. For the constant neighbourhood function (Table 2A), only nodes immediately adjacent in the x and/or y direction are included. To generate the discrete neighbourhood (Table 2C and Table 2D), the $h_{ci}(\tau)$ contracts to a predetermined neighbourhood for each iteration.

2.4.1 Batch Training Algorithm

The SOMs can be trained by two different algorithms: batch and sequential. These algorithms update the values of the nodal centres differently. The batch training algorithm (batch) updates the values of the nodal centres after all N data vectors are re-assigned to their BMU at each iteration τ of the learning/training phase of the SOM. The nodal centres are updated a total of T times. At each iteration τ, the nodal centres are updated using all the data vectors in the node of interest (BMU) and all vectors within the neighbourhood of the BMU. The value of the i^{th} nodal centre, $W_i(\tau+1)$, is computed by:

$$W_i(\tau+1) = \sum_{j=1}^{N} \frac{h_{ci}(\tau) \cdot V_j}{h_{ci}(\tau)} \qquad \textbf{[11]}$$

where V_j is the data vector j, N is the total number of data vectors, and the contribution (weight) of each vector to the nodal centre is determined by the Gaussian neighbourhood, $h_{ci}(\tau)$ (equation [7]).

2.4.2 Sequential Training Algorithm

The sequential training algorithm (seq) recomputes the values of the nodal centres after each data vector is re-assigned to its BMU during the iteration τ of the learning/training phase of the SOM. The new nodal centres for the sequentially trained algorithm are recomputed N times more often than for a batch training algorithm. When a data vector gets re-assigned to its BMU, the BMU and all nodes in its neighbourhood have the values of their nodal centres updated.[23,24] This is done by a weight vector adjustment at iteration τ:[23,24,30]

$$W_i(\tau + 1) = W_i(\tau) + \alpha(\tau) \cdot h_{ci}(\tau) \cdot (V_j - W_i(\tau)) \quad \textbf{[12]}$$

where $W_i(\tau)$ is the value of the nodal centre of node i at iteration τ, $W_i(\tau + 1)$ is the updated nodal centre i at iteration $\tau + 1$; V_j is the conformational vector j in the dataset, which has been *randomly* selected from the N members. In equation [12], $\alpha(\tau)$is the learning rate factor, and $h_{ci}(\tau)$ is the neighbourhood function, (equation [7]). For each iteration, τ, due to the random selection of V_j from the N members of the dataset, the sequential ordering in which the dataset is re-assigned to the BMUs is different.

2.4.2.1 Learning Rate Factor

For the sequential training algorithm, the second aspect which contributes to the change in the values of the nodal centre during SOM training is the learning rate factor, $\alpha(\tau)$ in equation [12]. This factor is similar to the neighbourhood radius function, η_τ, in that it decreases over the iterations $\tau = 0$ to T. Both the functions $\alpha(\tau)$ and η_τ contribute to the updates of the nodal centres that are computed via the sequentially trained algorithm, equation [12]. There are five different learning rate functions from which $\alpha(\tau)$ can be selected to train a SOM of dimensions ($\eta \times \eta$):[31]

CONST: $$\alpha(\tau) = \left(1 - \frac{0.95\tau}{T}\right) \quad \textbf{[13]}$$

EXP: $$\alpha(\tau) = \alpha_0 \cdot \exp\left(\frac{-\tau}{(T/\ln(\eta))}\right) \quad \textbf{[14]}$$

INV: $$\alpha(\tau) = \frac{\alpha_0}{1 + 100\tau/T} \quad \textbf{[15]}$$

LINEAR: $$\alpha(\tau) = \alpha_0\left(1 - {}^{\tau}/_{T}\right) \quad \textbf{[16]}$$

POWER: $$\alpha(\tau) = \alpha_0(0.005/\alpha_0)^{{}^{\tau}/_{T}} \quad \textbf{[17]}$$

where τ is the iteration number, T is the total number of iterations, and α_0 is the initial learning rate factor. For future reference, the functions are labelled with descriptors on the left. All these $\alpha(\tau)$ functions decrease over the number of iterations τ. Near the beginning of the training process, many data vectors will be re-assigned. As the training progresses, fewer data vectors are re-assigned until convergence is reached (no changes in nodal memberships).

2.5 Evaluation of SOMs

Two objective functions have been mentioned in Section 1.2. However, a third objective function can be implemented with an SOM, which pertains to the neighbourhood surrounding a cluster. SOMs cluster the data vectors based on intra nodal similarity, but SOMs also organize the nodal relationships on the 2-dimensional map. The data vectors assigned to nodes adjacent to each other or within the same neighbourhood are more similar to each other than to members of nodes farther away. Other partitional methods do not use this topological (neighbourhood) component in training the clusters. The NEIGH incorporates a description of the neighbouring nodal contents, as well as that of individual clusters.

$$\underset{(NEIGH)}{Objective\ Function} = \sum_{k=1}^{\eta^2} \sum_{m\ in\ N_k} \sum_{j=1}^{P_m} \left(V_j - W(\tau)_k\right)^2 \quad \textbf{[18]}$$

where N_k is the number of clusters in the neighbourhood of cluster k, P_m is the total number of vectors in the clusters surrounding cluster k, and $W(\tau)_k$ is the nodal centre of cluster k. The different objective functions were calculated after the SOM was trained. The objective functions INSSQ and BWTSSQ can compare clustering quality for bordered, toroidal and untrained maps. The objective function NEIGH was used to compare cluster quality for SOMs with the same boundary conditions (bordered to bordered or toroidal to toroidal). Since, the toroidal maps have more neighbouring nodes on the edge than the borders SOMs.

There have been few reports of SOMs being used to cluster macromolecular conformations obtained from computer simulations.[1–4] The limitation is that only a crude comparison between different SOMs trained using the same data is possible. For example, SOMs with identical nodal memberships can have those clusters located in a different place on the map; for example, the nodal memberships can be mirror images of each other. Two methods for SOM comparison have been proposed in the literature, topological error (TE)[32] and quantization error (QE).[32] These error measurements evaluate the overall performance of data clustering. TE measures the distortion of the dataset on the SOM. The BMU compared to the second best matching unit (SBMU) should be adjacent, where the SBMU is the node with the second smallest distance between the data vector and the nodal centre. In Table 2A the neighbourhood of the BMU (black) should contain the SBMU in one of the grey nodes. For the other neighbourhoods in Table 2, TE is calculated for the smallest neighbourhood.

$$TE = \frac{\sum_{i=1}^{N} \begin{matrix} 1 & if\ b_1 and\ b_2\ are\ not\ connected \\ 0 & if\ b_1 and\ b_2\ are\ connected \end{matrix}}{N} \quad \textbf{[19]}$$

where N is the number of conformations, b_1 is the BMU, b_2 is the SBMU.

QE is the measure of how similar the nodal centre is to the data vectors in the membership of each BMU:

$$QE = \frac{\sum_{i=1}^{N} D(W_j, V_i)}{N} \quad \textbf{[20]}$$

where $D(W_j, V_i)$ is the Euclidean distance between the j^{th} value of the nodal center, W_j, and the i^{th} data vector. $D(W_j, V_i) = 0$ if V_i is not a member of the j^{th} node.

The TE, QE and the different types of objective functions give a single quantitative value for each evaluation for a single map. TE gives an evaluation for the neighbourhood and QE gives an evaluation for one cluster. The objective functions (equations [1], [2] and [18]) can give either an evaluation of the clusters or an evaluation on the neighbourhood or a combination of the two. However, these values do not compare the results of independent partitionings. Different SOMs can produce similar values of TE, QE or objective functions but can have completely different partitionings. Other methods must be introduced to determine whether two maps are similar. The Cramérs V-index and similarity index have been used in the past to compare the partitioning of SOMs.[24,33] These methods can be used to help determine if SOMs can produce reproducible clusters.

2.5.1 Cramérs V-index

The Cramérs V-index, C_v, is a statistical test score used to describe the similarity between different partitions of the same dataset.[34] The C_v is widely used in social and psychological

sciences to determine how similar an independent sample is to another with the same and different sizes of groups.[35] The C_v value is used when there are more than two categories or the partitionings have a different number of categories (i.e. the number of clusters is different between the different partitionings). [35] The use of C_v is not new; however, our particular application of C_v to determine reproducible clusters is novel. The value of C_v is between 0 to 1, where 0 is when the data partitioning exhibits no similarity at all, and 1 is when data has been partitioned identically. The equations describing C_v below have been modified to describe its application to comparing two SOM partitionings of the same dataset, one with $\eta_a \times \eta_a$ nodes and the other with $\eta_b \times \eta_b$ nodes.

The C_v statistic depends on the chi-squared value, χ^2: [20,33]

$$C_v = \sqrt{\frac{\sum_{i=1}^{{\eta_a}^2} \sum_{j=1}^{{\eta_b}^2} {\chi^2}_{ij}}{N \cdot \min({\eta_a}^2 - 1, {\eta_b}^2 - 1)}} \quad \textbf{[21]}$$

where

$${\chi^2}_{ij} = \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \quad \textbf{[22]}$$

and

$$E_{ij} = \frac{n_i n_j}{N} \quad \textbf{[23]}$$

In equations [21], [22] and [23], n_i is the number of identical members of node i found in all other nodes of map b, n_j is the number of identical members of node j found in all the other nodes of map a, N is the total number of data vectors being clustered on the maps. The observed similarity, O_{ij}, is that between the node i on map a compared to node j on map b, and E_{ij} is the expected value of the comparison of node i to node j. Equations [21] to [23] can be manipulated

to compare only the interior nodes from other SOMs to evaluate the similarity between different boundary conditions (bordered and toroidal).

The χ^2 value validates the differences between the different partitionings of the dataset by SOMs a and b and determines if the partitioning is due to chance or to variance in the clustering.[33] One advantage of examination of individual values of ${\chi^2}_{ij}$ is that they can be used to relate the nodes of one independently trained map to the nodes of another to locate similar clusters placed in different nodal locations. When the value of C_v is high and the maps have the same number of nodes, a 1:1 relationship between nodal memberships on both maps can be located. When the value of C_v is low, the nodal memberships matched in a 1:1 relationship is less likely, since nodes from one map can be split up on the other map because of the partitioning is different.

Table 3 is an example of the comparison between two 2×2 SOMs with 20 vectors on each map. The columns are the memberships from SOM a and the rows are members from SOM b. The numbers on the upper table of Table 3 are the observed values O_{ij}, which is the number of shared members between SOMs a and b in nodes i and j. For example the $i = 1$ and the $j = 1$ nodes share three identical members. The value for n_i represents the shared membership of node i on map a with all nodes of map b. The sum of $n_i = n_j = N = 20$ which is the total number of vectors. The ${\chi^2}_{ij}$ (the bottom of Table 3) are computed by equation [22] and E_{ij} is from equation [23] with the appropriate values. The C_v (equation [21]) value is low, since it is closer to zero than one. However, a comparison can still be made to show the similarity between partitioning of the data on maps a and b. Data in node $j = 1$ from map b is divided between two nodes on map

a (nodes $i = 1$ and $i = 3$). Both these nodes have higher ${\chi^2}_{ij}$ values than any other node on the map (${\chi^2}_{1\,1} = 6.75$, ${\chi^2}_{3\,1} = 4.50$).

Table 3: An example of how ${\chi^2}_{ij}$ is used to relate SOM nodes on different SOMs. Map a and b are 2×2 SOMs. The SOMs each contain the same 20 vectors.

O_{ij}		i				n_j
		a(1,1) $i=1$	a(1,2) $i=2$	a(2,1) $i=3$	a(2,2) $i=4$	
j	b(1,1) $j=1$	3	0	2	0	5
	b(1,2) $j=2$	0	2	0	3	5
	b(2,1) $j=3$	0	2	0	3	5
	b(2,2) $j=4$	0	0	0	5	5
n_i		3	4	2	11	20

${\chi^2}_{ij}$		i				
		a(1,1) $i=1$	a(1,2) $i=2$	a(2,1) $i=3$	a(2,2) $i=4$	
j	b(1,1) $j=1$	6.75	1.00	4.50	2.75	
	b(1,2) $j=2$	0.75	1.00	0.50	0.02	$C_v = 0.39$
	b(2,1) $j=3$	0.75	1.00	0.50	0.02	
	b(2,2) $j=4$	0.75	1.00	0.50	1.84	

Lisboa *et al.* used the k-means algorithm to cluster a synthetic dataset and a large bioinformatics dataset.[20] The quantitative assessment used in most k-means algorithms is within-group-sum-of-squares (INSSQ; equation [1]). To assist the comparison of different partitioning of the same datasets, the C_v was used to show the similarity of the clusters. The bioinformatics dataset was a cardiotocography dataset comprised of 2126 data vectors and it is reported that it is hard to get reproducible clusters.[20] The median C_v value was used in combination with the

INSSQ to determine the most reproducible clusters, and the optimum number of clusters. To find reproducible results, the k-means algorithm was run 500 times for each total number of clusters. The total number of clusters ranged from two to fifteen. The authors found that the median C_v value is high until the optimal number of clusters is reached; as the number of clusters is increased beyond that, the median C_v values start to decrease. The combination of these two measures and running the algorithm multiple times ensured that stable and reproducible clusters were found.

Garge *et al.* used C_v values to quantify the stability of clusters generated from four different iterative partitioning algorithms (forming *k* clusters, where *k* = 2 to 10), including k-means, fuzzy clustering, and SOM (1×*k*).[33] The clustering algorithms clustered the same 37 real microarray datasets and eight simulated microarray datasets.[33] The datasets contained between 12 to 1200 data vectors. The results for all clustering algorithms resulted in increased C_v values when larger datasets where used. This is the only literature example where C_v values were used in the analysis of SOMs. However, the SOMs used in Garge *et al.* are small with a maximum number of nodes of 10. This was done so that all clustering algorithms could be compared on the same scale (number of clusters). One of the advantages of SOMs is the ability to partition data into a large number of clusters. In this work, SOMs have more than 25 nodes and C_v values are used to compare the reproducibility of the maps.

2.5.2 Similarity Index

The similarity index, SI, computes how similar two SOMs are to each other and gives a normalized value between zero and one,[24] where zero represents two entirely different partitionings of the same dataset and one describes two identical divisions of the data vectors into groups. For two SOMs, i and j, both of mapsize η×η, the similarity index is:

$$SI = \sum_{i=1}^{\eta^2} \sum_{j=i}^{\eta^2} \frac{{NN_{ij}}^2}{(NN_i)(NN_j)} \quad \textbf{[24]}$$

where NN_i are the number of pairs of data vectors placed in the same node of map i, NN_j are the number of pairs of data vectors placed in the same node of map j and NN_{ij} are the number of the same pairs of data vectors placed together in both map i and map j. A single node containing R data vectors, has R(R-1)/2 unique pairs in that node. Equation [24] evaluates similarity between cluster memberships of two different maps without consideration of the neighbourhoods used to make that map.

This work introduces the revised similarity index (rSI), and uses the same concept as the similarity index in equation [24] but also incorporates the neighbourhood. The pairs are not only data vectors in the BMU but also the second best matching unit (SBMU), if the second best matching unit is within the neighbourhood of the BMU (Figure 6). It is reasonable to assume that if a pair of closely-related data vectors are not placed in the same node (i.e. they each have the same BMU), then they each may appear in the membership of neighbouring nodes:

$$rSI = \sum_{i=1}^{\eta^2} \sum_{j=i}^{\eta^2} \frac{{NN_{ij}}^2}{(NN_i)(NN_j)} \quad \textbf{[25]}$$

where NN_{ij} are the number of the same pairs of data vectors placed together in either the BMU or the SBMU in both map i and map j. Figure 6 shows when data vectors are considered paired or not. If the SBMU is not within the neighbourhood of the BMU, the pairs contributing to NN_i or NN_j are only those of the BMU. If the SBMU is within the neighbourhood, the pairs contributing to NN_i or NN_j are both from the BMU and SBMU.

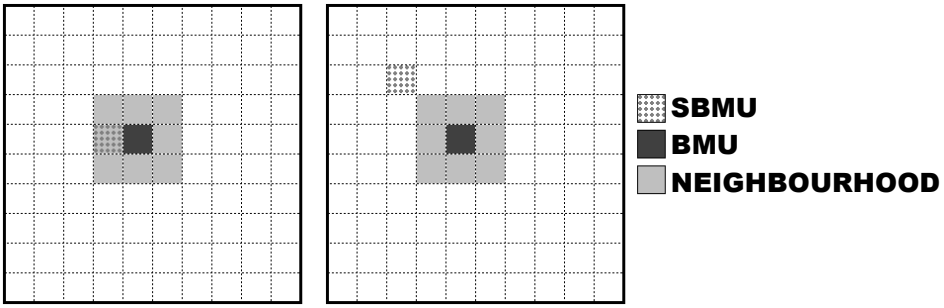


Figure 6: Graphical representation for considering pairs of data vectors in computation of the revised similarity index (rSI). A data vector has a BMU (dark grey) and a second best mating unit (SBMU) (dotted grey). If the SBMU is within the smallest neighbourhood of the BMU (left diagram) all data in these two nodes are considered paired. If the SBMU is not within the smallest neighbourhood (right diagram) the data vector is only paired with data from the BMU.

2.6 Clustering of Molecular Conformations Obtained by MD Simulations using SOMs

As previously mentioned in Section 1.4, MD conformational ensembles can be thought of as a dataset of vectors with n-dimensions. There is one vector for each conformation and a vector can contain values for dihedral angles or atomic positions in three-dimensional space to describe the protein. Since the main conformational change happens in the backbone, the Cartesian coordinates (x, y, z) for the position of the backbone atoms or the dihedral angles (φ, ψ) are often used. Sometimes only the positions of α-carbons are used to reduce the number of variables. These descriptors of the conformational ensemble can be used to cluster the protein conformational distribution obtained by MD simulations.

Hyvönen *et al.* used the SOM software in MATLAB[36] to cluster 1.44 million conformations obtained from MD simulations of a 1-palmitoyl- 2-linoleoyl-*sn*-glycero-3-phosphatidylcholine (PLPC) phospholipid membrane assembly on one SOM.[4] An individual conformation of PLPC was described by 41-dimensions since there are 41-dihedral angles contained in the lipid. A subset of the 41-dihedral angles, designated sn-2, was described by 18-dimensions. The PLPC conformations were clustered on two different bordered 10×10 hexagonal SOMs. These maps were initialized by maxmin. Two different clusterings of the phospholipid

were with the complete PLPC (41-dimensions) and the subset *sn*-2 (18-dimensions). The two different SOMs found all major structural features, including both conformations of the sn-2 region (straight and bent).

Murtola *et al.* used different mapsizes (48×72, 40×60, 32×48, 24×36, 16×24, and 8×12) of a bordered hexagonal SOM with the MATLAB software[36] to cluster conformations of PLPC from an MD simulation.[37] The SOM clustered 400 000 conformations of PLPC, which were described by 12-dihedral angles, generating a 12-dimensional vector. They used a discrete neighbourhood radius function and three different learning rate functions (INV, LINEAR, and POWER), along with an initial learning rate factor between 0.1 and 0.5 for the linear function. They determined that the larger map (48×72) was more useful for describing the lipid conformational clusters, as well as the power and linear learning rate functions, where the linear function used an initial learning rate of 0.3. They found that a training length between 5 and 10 was appropriate to train the SOM. Since there was so many clusters, a secondary clustering method (hierarchical agglomerative clustering) was used to reduce the number of nodes. They were able to use this method to give a qualitative understanding of the structural regions of the lipid by an un-supervised learning algorithm. The main advantages they found was that a large dataset could be reduced to a few relevant PLPC conformations to determine the important features of the system (the nodal centres were used). They proposed that SOM clustering would be a good starting point for conformational analysis for biomolecules.

Fraccalvieri *et al.* clustered 40 000 conformations of the α-spectrin SH3 protein domain and six mutants based on 55 α-carbon Cartesian coordinates (165-dimensional data: 55×3=165) obtained from MD simulations.[38] The SOM was optimized for its parameters (Training length= 5000, Neighbourhood Function = Gaussian, Mapsize = 10×10). Once an SOM was trained with a

combination of mutant or wild type datasets (cold-adapted α-amylase to that of the mesophilic counterpart (warm-adapted)), the SOM clusters were grouped together by a hierarchical algorithm to determine if the mutants and the wild type explored the same conformational space. This reduced the number of conformations which needed to be examined by resulting in, at most, five clusters.[16] They determined specific dynamic structures to relate these seven proteins (six mutants to wild type), i.e. to guide the design of a mesophilic-like mutant from the cold-adapted α-amylase.

Bouvier *et al.* clustered the 50 000 conformations obtained by MD simulations of a modified vancomycin (VanA and $VanA_{ss}$; Cys52 and 64 form a disulfide bridge) ligase onto a single 50×50 rectangular toroidal SOM with a discrete neighbourhood radius function. The learning rate function was POWER and an initial learning rate factor of 0.5 was employed.[39] The study was focused on the ω-loop, which is thought to play a key role in substrate binding. The 343 inter-α-carbon Cartesian coordinate distances, transformed into the covariance matrix, were used to cluster these conformations onto the SOM. Once the SOM was generated, a distance matrix was used to group the members of nodes together. The SOM revealed four distinct conformational basins for the enzyme to interact with the ligands. VanA appears to have high level of resistance to the modified precursor of D-Ala-D-Lac, which can be the target for the design of new antibacterials. The SOM was able to distinguish between ligand functional classes. The SOM was able to give representative structures that have different binding properties with the slight opening of both VanA and $VanA_{ss}$. The representative structures generated by the second clustering method were used as target receptor conformations to which key ligands involved in the ligase mechanism were docked in computational docking experiments. This work

shows how SOMs have been used to classify biomolecular structures and to provide a protocol for design of novel inhibitors.

Many of these examples used SOMs as the first training phase and used a secondary method to narrow down the number of relevant structures for examination. Furthermore, many of these experiments did not evaluate the reproducibility of the SOM(s) produced. In this work, SOMs are the only method used to cluster protein and peptide conformations.

2.7 SOM Programs and Algorithms

The SOMs explored in this work were generated from two independent source codes. The first program was a C++ version of a Kohonen map,[40,41] modified to include three different initializations: maxmin, rand, and rvec, the implementation of toroidal boundaries, a decreasing neighbourhood function and an exponential decreasing learning rate factor. The constant neighbourhood function and the corresponding learning rate factor were original to the program; these parameters were only used for the initial assessment and the program was altered to enable use of a decreasing neighbourhood.[42] For a constant neighbourhood algorithm, the change of the values of nodal centres is limited by the current membership of each node and its immediate neighbourhood and this can cause an increase in the training length required for convergence.[43]

The second SOM program used for the analysis was *somtoolbox* 2.0 in MATLAB,[31,36] in which the toroidal boundary algorithm was modified. In order to have a completely hexagonal SOM, the mapsize must be an even number; if the size of the map is an odd number, the map has both rectangular and hexagonal characteristics. Both programs used a Gaussian neighbourhood function and could implement both bordered and toroidal boundaries. Table 4 is a list of the other parameters which were used in the programs.

Table 4: A list of parameters the C++ and MATLAB SOM programs can implement.

	Nodal Geometries[a]	Initialization[b]	$dist$[a]	Training Algorithm[c]	η_τ[c]	$\alpha(\tau)$[d]	T[c]
C++	RECT	maxmin rand rvec	Euclidian	Seq[12]	CONST NEIGH[8] EXP ↓NEIGH[9]	CONST[13] EXP[14]	100 500 1000 5000 10000 15000
MATLAB	RECT HEXA	maxmin	Lookup table	Seq[12] Batch[11]	DISC ↓NEIGH[10]	INV[15] LINEAR[16] POWER[17]	100 500 1000 5000 10000 15000 20000

[a] description can be found in section 2.1
[b] description can be found in section 2.3
[c] description can be found in section 2.4, square brackets refer to the equation numbers
[d] description can be found in section 2.4.2.1, square brackets refer to the equation numbers

3 Computer Simulations of Molecules

One type of molecular computer simulations is Molecular Dynamics (MD) simulations. MD simulations represent a simple and efficient method to understand the microscopic origin of physical properties or to predict the expected behaviour of a system, which is not accessible with experiment. As mentioned before in section 1.3.1, if an MD simulation is run for an infinite amount of time all conformational space will be explored by the molecular system under investigation. Since this is not feasible, often multiple simulations are run from different starting conformations. If the ensembles explored overlap to a significant degree, a sufficient amount of conformational space is considered to have been explored.

Force fields are used in conjunction with MD software to replace the true potential of the system with a simplified version, assumed to be valid under the conditions being simulated. The reliability and accuracy of the simulation is dependent on the force field. The force field parameters are typically obtained via quantum mechanical calculations or experimental data. The different types of software treat the parameter values a bit differently (e.g. atomic charges cannot

be determined by experiment and the method for determining the atomic partial charges vary). Furthermore, the treatment of solvent (water), as well as the truncation of the long range electrostatics interactions can vary. Different simulation software programs with their different force fields have different strengths and weaknesses. However, the choice of which force field to use is dependent on the problem being studied.[44]

3.1 The CHARMM Force Field

"Force field" is the conventional term to describe the potential energy function used in molecular-mechanics-based Molecular Dynamics (MD) simulations. The potential energy function employs simple empirical energy functions[45] that are dependent on the system's atomic coordinates, corresponding to a particular molecular (protein) conformation.[46] The goal of MD simulations is to make empirical approximations to generate an ensemble (collection or sample) of protein conformations which follows the Boltzmann distribution of state. The approximations used to generate the protein conformational ensemble are the equations used to estimate the potential energy. At interatomic separations close to the empirically determined equilibrium distances, the approximations are relatively accurate. However, as the atoms move further away from their equilibrium distances, the potential energy calculated does not mimic experimental results. For example the potential energy of a covalent bond between two atoms is simulated by a quadratic function. However, this does not account for the energy allowed to break a bond (no longer quadratic) since eventually they are just separate atoms.

Chemistry at Harvard Macromolecular Mechanics (CHARMM) is both a force field and a molecular simulation program originally designed to study proteins.[47] CHARMM uses the assumption that the atom is the fundamental unit, which is considered a sphere with an

embedded point charge.[46,47] The point charge is the (fixed) partial charge on the atom. Therefore, molecular systems are described without electrons.

The CHARMM potential energy is calculated from six terms: bond energy, angle energy, dihedral energy, improper angle energy, electrostatic potential, and van der Waals interactions. All of these potential energy terms are dependent on the Cartesian coordinates of the atoms:[46,48]

$$U(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = \sum_{\substack{ALL \\ BOND}} U_{BOND}(b) + \sum_{\substack{ALL \\ ANGLE}} U_{ANGLE}(\theta) + \sum_{\substack{ALL \\ DIHED}} U_{DIHED}(\varphi) + \sum_{\substack{ALL \\ IMPRO}} U_{IMPRO}(\omega) + \sum_{\substack{ALL \\ ELEC}} U_{ELEC}(r_{ij}) + \sum_{\substack{ALL \\ VDW}} U_{VDW}(r_{ij}) \quad \textbf{[26]}$$

where the vector $\vec{r}_i$ describes the x, y and z coordinates of atom i and N is the total number of atoms in the system.

The bond potential energy, U_{BOND}, is due to bond stretching and contracting of atoms covalently bonded together, and is represented by a harmonic function: [46,48]

$$U_{BOND}(b) = k_b(b - b_0)^2 \quad \textbf{[27]}$$

where k_b represents the force constant in kcal/(molÅ^2), b_0 represents the equilibrium bond distance between two atoms in Å, and b is the bond distance calculated from the atomic coordinate system.[46,48] The parameters for a particular pair of covalently bonded atoms are k_b and b_0.

The angle potential energy, U_{ANGLE}, is due to angle bending. This is represented by a harmonic function: [46,48]

$$U_{ANGLE}(\theta) = k_\theta(\theta - \theta_0)^2 \quad \textbf{[28]}$$

where k_θ represents the force constant in kcal/(mol degrees^2), θ_0 represents the equilibrium bond angle between three atoms in degrees, and θ is the bond angle calculated from the atomic coordinates. The parameters for a particular bond angle formed between three atoms are k_θ and θ_0.[46,47]

The dihedral angle potential energy, U_{DIHED}, is due to the torsion angle potential energy function, which is dependent upon the hybridization between bonded atoms.[46] The dihedral angle is computed from the location of four sequentially bonded atoms where the two central atoms are defined as the axis of rotation.[47] The dihedral energy is represented by a sum of sinusoidal functions:[47,48]

$$U_{DIHED}(\varphi) = \sum_{n=1}^{n_{max}} k_\varphi(1 + \cos(n\varphi - p)) \quad \textbf{[29]}$$

where k_φ represents the force constant in kcal/mol, the integer n_{max} represents the number of components in the Fourier series, n represents the periodicity of the dihedral angle φ (i.e. controls the number of energy minima as the dihedral angle is rotated between 0° and 360°)[47] and p is the phase shift in degrees. The phase shift is used to control the location of energy minima for the corresponding dihedral angle, φ.

The improper dihedral angle potential energy, U_{IMPRO}, is meant to maintain planar and chiral conformations. The improper dihedral angles are computed from the location of four atoms, one being the central atom, X, with the other three atoms a, b, and c bonded to the central atom (Figure 7). The improper dihedral angle is the angle formed between two planes aXb and bXc (Figure 7).

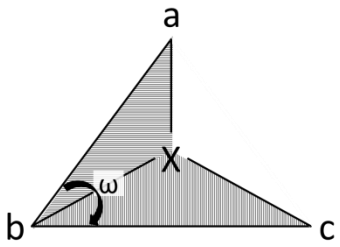


Figure 7: Definition of an improper dihedral angle, ω.

The potential energy of the improper dihedral can be mathematically represented as a harmonic function:[46]

$$U_{IMPRO}(\omega) = k_\omega(\omega - \omega_0)^2 \quad \textbf{[30]}$$

where k_ω represents the force constant in kcal/(mol degrees^2), ω_0 represents the desired angle in degrees, where the central atom is in a planar conformation or appropriate chiral configuration, and ω is the actual improper angle computed from the atomic coordinates. The parameters for a particular improper angle between four atoms are k_ω and ω_0.

The potential energy of non-bonding interactions includes the electrostatic interactions and the van der Waals interactions. The interactions will only be calculated if the atoms are separated by more than three covalent bonds and if the distance is smaller than a user specified cutoff distance. The potential energy between two atoms i and j due to electrostatic interactions, $U_{ELEC}(r_{ij})$, is calculated by Coulomb's Law:[11]

$$U_{ELEC}(r_{ij}) = \frac{1}{4\pi\epsilon_0}\left(\frac{q_i q_j}{\varepsilon r_{ij}}\right) \quad \textbf{[31]}$$

where $q_i$and q_j represent the partial point charges of atoms $i$and j in elementary charge units, ε represents the relative dielectric constant of the surroundings, ϵ_0 represents the relative dielectric constant in a vacuum (permittivity of free space) and r_{ij} represents the interatomic distance in units of Å. The dielectric constant, ε, can be used to simulate the presence of solvent implicitly.

The dielectric constant simulates the solvent in the environment by screening the charges on atoms in the protein from each other, therefore affecting the electrostatics. An alternative is to explicitly model the solvent as discrete water molecules; when this is done, ε is set to 1. However, to have water explicitly present for large proteins is computationally expensive, since the motion of each water molecule as well as the atoms of the protein must be modelled. The more atoms in the simulation, the longer and more expensive the simulation will be. One of the ways to reduce computational time is to use a rigid point charge module, such as TIP3P, as a water model (Section 3.2), rather than a fully flexible (bond stretching and bond angle bending) model requiring intramolecular interactions to be computed.

The potential energy due to the van der Waals interaction, U_{VDW}, between two atoms *i* and *j* is calculated by the Lennard-Jones inverse 12-6 potential:[46]

$$U_{VDW}(r_{ij}) = 4\varepsilon_{ij}\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \quad \textbf{[32]}$$

where ε_{ij} represents the well-depth in kcal/mol, σ represents the sum of the van der Waals radii in Å, and r_{ij} is the interatomic distance in Å. When the U_{VDW} potential energy is positive, the first term of equation [32] dominates the expression, indicating steric overlap. When the potential energy is negative, the second term of equation [32] dominates the expression, representing dispersion forces.

In the implementation of the CHARMM force field program, the information used to calculate the potential energy of a protein is divided into two files: the parameter structure file (PSF) and the parameter file.[46,47] The PSF gives information about the atoms of the molecular system under investigation. This information includes the name of each atom, the atoms'

locations in Cartesian coordinates, to which amino acid each atom belongs in the primary amino acid sequence, the atomic masses, the partial point charges on the atoms, and the connectivity (i.e. bonds, angles, dihedral angles, and improper dihedral angles). The parameter file contains the empirical values of $k_b, b_0, k_\theta, \theta_0, k_\varphi, n, p, k_\omega, \omega_0, \varepsilon_{ij}$, and σ of equations [27]-[32].

3.2 TIP3P Water

The TIP3P water model is used with the CHARMM force field to simulate water as a molecular solvent.[49] The model TIP3P is a three-site model that corresponds to the three atoms in water; it is simulated by a sphere with an oxygen atom at the centre and two hydrogens attached to the oxygen, which are all contained within the sphere (Figure 8). The bonds, r_{OH}, and angle, θ_{HOH}, of water are held rigid or fixed at 0.9572Å and 104.52°, respectively.[49] Only the non-bonded electrostatic and Lennard-Jones potential are calculated between pairs of TIP3P waters:[49]

$$U_{ab} = 4\varepsilon_{ab}\left[\left(\frac{\sigma_{OO}}{r_{OO}}\right)^{12} - \left(\frac{\sigma_{OO}}{r_{OO}}\right)^{6}\right] + \sum_{i=1}^{3}\sum_{j=1}^{3}\frac{q_{ai}q_{bj}e^2}{r_{aibj}} \qquad \textbf{[33]}$$

where $q_i e$ is the charge on atom i and $q_j e$ is the charge on atom j, r_{OO} is the distance between oxygen atoms of monomers of waters a and b, and ε_{ab} represents the well-depth between oxygen atoms of monomers of waters a and b in kcal/mol. The charge is either 0.417e for hydrogen or -0.834e for oxygen.[49] The part after the addition sign is the electrostatic potential energy and part in front of the addition sign is the Lennard-Jones potential. The Lennard-Jones potential from equation [33] is zero when the water monomers are touching, that is when $r_{OO} = \sigma_{OO} = 3.5364$ Å.[49]

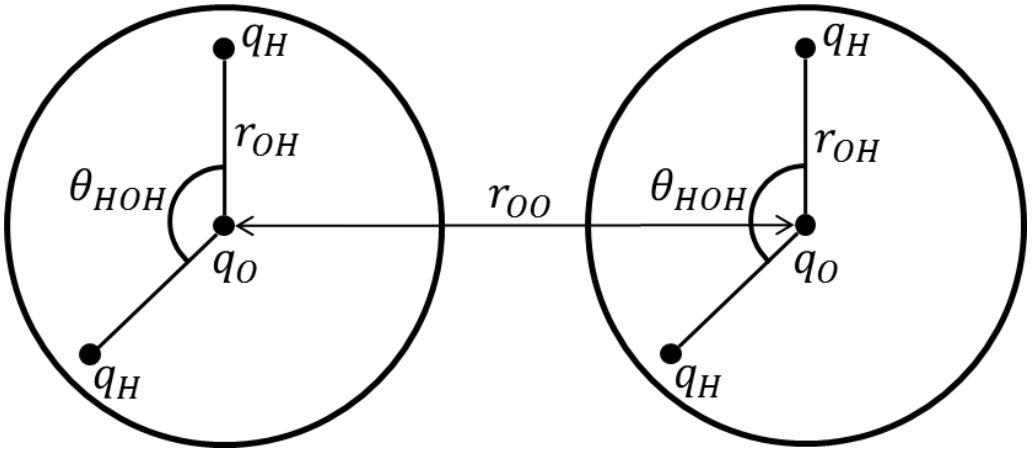


Figure 8: Diagram of the TIP3P water model.

3.3 NAMD

Nanoscale Molecular Dynamics (NAMD) is a simulation program developed for the study of large bimolecular systems,[48] and can be used with the CHARMM potential energy functions, parameters and file formats.[46] Molecular dynamics (MD) simulation is a computational approach to modelling the physical movement of atoms or molecules for a specified period of time. This is determined by numerical integration of Newton's equations of motion.[48] MD simulations predict the physical movement of atoms over a series of small time steps.

3.3.1 Energy Minimization

MD simulations may be preceded by a short energy minimization. Energy minimization is the process of finding a local minimum in the potential energy function. Prior to commencing a molecular dynamics simulation, a short energy minimization is often used to relieve unfavorable interactions that would give rise to unrealistic forces and instability in the MD simulation, resulting in large movements in the atom positions. In NAMD, the algorithm used for energy minimization is the conjugate gradient method.[48] The root mean squared gradient,

G_{RMS}, is computed as a scalar using the forces, $\vec{F}$, in the x, y and z directions, for all N atoms obtained from the potential energy function, equation [26]:

$$G_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\vec{F}_{x_i}\right)^2 + \left(\vec{F}_{y_i}\right)^2 + \left(\vec{F}_{z_i}\right)^2} \quad \textbf{[34]}$$

When the gradient is 0 kcal/molÅ, the system is at an energy minimum. Prior to starting a molecular dynamics simulation, unfavourable interactions need only be reduced, therefore a true energy minimum need not be obtained and the gradient does not need to reach 0 kcal/molÅ.

The forces on the individual potential energy terms are calculated by finding the first derivatives of the components of the six potential energy equations [26] with respect to x, y and z directions. Equations [35] and equation [36] show the partial derivative $\vec{F}_{x_i}$ on atom i in the x direction and equation [37] shows the force on an individual atom i;

$$\vec{F}_{x_i} = -\frac{\partial U_i(b,\theta,\varphi,\omega,r_i)}{\partial \vec{x}_i} \quad \textbf{[35]}$$

$$\begin{aligned}\vec{F}_{x_i} = &-\sum_{\substack{ALL\\BONDS\\TO\ i}} \frac{\partial U_{BOND_i}(b)}{\partial b}\frac{\partial b}{\partial \vec{x}_i} - \sum_{\substack{ALL\\ANGLES\\TO\ i}} \frac{\partial U_{ANGLE_i}(\theta)}{\partial \theta}\frac{\partial \theta}{\partial \vec{x}_i} \\ &- \sum_{\substack{ALL\\DIHEDS\\TO\ i}} \frac{\partial U_{DIHED_i}(\varphi)}{\partial \varphi}\frac{\partial \varphi}{\partial \vec{x}_i} - \sum_{\substack{ALL\\IMPROS\\TO\ i}} \frac{\partial U_{IMPRO_i}(\omega)}{\partial \omega}\frac{\partial \omega}{\partial \vec{x}_i} \\ &- \sum_{\substack{ALL\\ELECS\\TO\ i}} \frac{\partial U_{ELEC_i}(r_{ij})}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial \vec{x}_i} - \sum_{\substack{ALL\\VDWS\\TO\ i}} \frac{\partial U_{VDW_i}(r_{ij})}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial \vec{x}_i}\end{aligned} \quad \textbf{[36]}$$

$$\vec{F}_i = \vec{F}_{x_i}\hat{x} + \vec{F}_{y_i}\hat{y} + \vec{F}_{z_i}\hat{z} \quad \textbf{[37]}$$

where the force is dependent on the Cartesian coordinates of the atoms, $\frac{\partial U_i(b,\theta,\varphi,\omega,r_{ij})}{\partial \vec{x}_i}$ is the partial derivative of the function with respect to $\vec{x}_i$. The vector force in the x direction is $\vec{F}_{x_i}$. The total force on atom i is $\vec{F}_i$.

The conjugate gradient method in NAMD works by moving the atoms small distances in the direction of the calculated forces, for a user-supplied number of iterations or steps.

3.3.2 Classical Molecular Dynamics Simulations

In MD simulations, Newton's equations are applied in order to follow the motions of atoms with respect to time. Since it is impossible to solve these analytically for a multi-atomic system, a numerical integration is performed.[48] The positions of atoms are predicted at discrete time intervals t, $t + \delta t$, $t + 2\delta t$, …, where δt is a short time step.The time step, δt, must be smaller than the fastest atomic motion. In molecules, this is the vibrational motion of covalently bonded hydrogens, which is on the order of 1 fs.[46,47]

NAMD calculates the trajectories of atoms by an integration algorithm called Velocity Verlet.[13] This algorithm has an error of $O(\delta t^2)$.[48] Newton's equations of motion are solved in the order of equation [36], equation [37] and then equation [38], to give acceleration, $\vec{a}_{t_i}$, at time t for atom i.[48] The acceleration is used in equation [39] with the initial velocity, $\vec{v}_{t_i}$, at time t of atom i and to give the velocity of the system at half the time step, $\vec{v}_{\left(t+\frac{\delta t}{2}\right)_i}$.[48] To solve for the new atomic coordinates, $\vec{r}_{(t+\delta t)_i}$ of atom i, equation [40] is used along with the velocity at the half

time step equation [39].The new atomic velocity at time $t + \delta t$, $\vec{v}_{(t+\delta t)_i}$, is then solved by equation [41].[48]

$$\vec{F}_i = m_i \vec{a}_{t_i} \quad \textbf{[38]}$$

$$\vec{v}_{\left(t+\frac{\delta t}{2}\right)_i} = \vec{v}_{t_i} + \left(\frac{\delta t}{2}\right)\vec{a}_{t_i} \quad \textbf{[39]}$$

$$\vec{r}_{(t+\delta t)_i} = \vec{r}_{t_i} + (\delta t)\vec{v}_{\left(t+\frac{\delta t}{2}\right)_i} \quad \textbf{[40]}$$

$$\vec{v}_{(t+\delta t)_i} = \vec{v}_{\left(t+\frac{\delta t}{2}\right)_i} + \left(\frac{\delta t}{2}\right)\vec{a}_{t_i} \quad \textbf{[41]}$$

In equations [38]-[41], $\vec{F}_i$ is the force on atom i in the simulation obtained from equation [37], m_i is the atomic mass of the i^{th} atom, $\vec{a}_{t_i}$is the acceleration of the atom i at time t, $\vec{v}_i$ is the velocity of the i^{th} atom at a specific time indicated by the subscript, and $\vec{r}_i$ is the position of the i^{th} atom at a specific time indicated by the subscript.

The assumption in the Velocity Verlet method is that the acceleration is constant for the length of the small time step, δt.The time step is the limiting variable because $\vec{a}_t$ can only be assumed constant for a short time step.[46,48]

The atomic velocities in equation [39] are initialized from a random assignment from the Maxwell-Boltzmann distribution:[50]

$$P(\vec{v}) = \left(\frac{m_i}{2\pi k_B T}\right)^{\frac{3}{2}} 4\pi {\vec{v}_{0_i}}^2 \exp\left(\frac{-m_i {\vec{v}_{0_i}}^2}{2k_B T}\right) \quad \textbf{[42]}$$

where $\vec{v}_{0_i}$is the initial velocity being assigned to an atom i, k_B is the Boltzmann constant and T is the temperature of the system in K. The Maxwell-Boltzmann distribution is dependent on

temperature. The higher the temperature, the faster an object can move. Therefore the velocities are dependent on the temperature of the system.

It is common to rescale velocities at set intervals, in order to "heat" the system gradually to a desired temperature or to adjust for incremental numerical errors within the integration scheme. One method used is:

$$\vec{v} = \vec{v}_j \sqrt{\frac{T_0}{T_{ave}}} \quad \textbf{[43]}$$

where $\vec{v}$ is the new velocity, $\vec{v}_j$ is the current velocity of the j^{th}atom, T_0 is the desired temperature of the system and T_{ave} is the average temperature since the last rescaling. This method of rescaling shifts all the velocities by the same factor. Another method of ensuring that the atomic velocities reflect the desired temperature T_0 is to use Langevin dynamics, discussed in Section 3.3.2.5.

3.3.2.1 SHAKE

The SHAKE algorithm can be used to increase the size of δt in a MD simulation.[46,47,51] This is desirable since hydrogen is the fastest moving atom. Constraining the bond lengths of covalently bonded hydrogens, means that hydrogen is now not the fastest moving atom. This is done by Lagrangian Multipliers. Using SHAKE, δt can be increased two-fold to 2 fs.[46,51]

3.3.2.2 Periodic Boundary Conditions

Periodic Boundary Conditions (PBC) are a way of representing an infinitely large system.[52] Figure 9 is an example of PBC. The simulation cell has permeable walls; however, the number of atoms is constant.[52] When an atom exits the unit cell it appears on the opposite side with the same velocity.[52] The grey box in the centre of Figure 9 is the simulation cell; the eight

boxes surrounding it are identical to the simulation cell. When an atom exits the central simulation cell, it is replaced by an image atom from one of the other surrounding boxes. Therefore the number of atoms in the central simulation cell remains constant.[52] The simulation cell used in this work is cubic. However, other geometries can be used. The geometries must be repeating units and no space can be found between image cells (e.g. truncated octahedral).

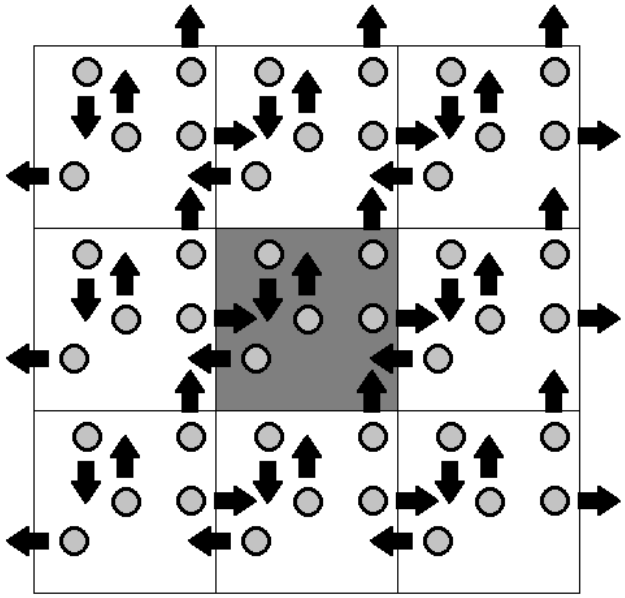


Figure 9: Representation of periodic boundary conditions in two-dimensions. The circles are the atoms; the black arrows are the atomic velocities.

3.3.2.3 Particle Mesh Ewald

NAMD uses a cutoff distance when computing both the Lennard-Jones and electrostatic potentials to reduce the computational time.[48,53] A cutoff distance is a maximum interatomic distance over which non-bonded energies can be computed. The cutoff distance is chosen so atoms in the centre simulation cell do not "see" (interact with) their images in any of the surrounding cells. With larger simulation cells, the cutoff distance can also be increased. For example with a cubic cell of length L the maximum cutoff distance can be L/2. However, the increase in the cutoff distance increases the simulation time since more non-bonded terms must

be calculated. The implementation of the cutoff distance results in a truncation of the non-bonded potential energy terms.[48,53] To avoid truncation of the non-bonded potential energy function at the cutoff distance, a smooth switching function can be incorporated that smoothes the function to zero starting at the switching distance. The Particle Mesh Ewald (PME) estimates the long range electrostatic interactions omitted by the implementation of the cutoff distance. [52,53] The PME algorithm splits the electrostatic energy into reciprocal Ewald sums.[53] An Ewald sum is the solution to Poisson's equation in periodic boundary conditions with Gaussian charge densities as sources.[48,54] In essence, the electrostatic interactions of atoms within the central simulation cell and the periodic images of the surrounding PBC cells are treated as a (convergent) lattice sum. To reduce the computational time, a pair-list distance can be generated. This pair list distance is a list of atoms that should be checked before the energy is calculated, to check if any atoms ventured into the cutoff distance.

3.3.2.4 Types of Ensembles

There are different types of ensembles that can be used in a molecule simulation. The ensembles have fixed state properties; this means certain properties are held constant by algorithms in the simulation. The three different simulations used in this work are NVE, NPT, and NVT. The NVE ensemble is called a microcanonical ensemble, where the total energy, volume and the number of atoms of the system are held constant. The NPT ensemble or isothermal-isobaric ensemble maintains a constant number of atoms, pressure and temperature. In an NPT ensemble, the volume is allowed to fluctuate. The NVT ensemble or Canonical ensemble is where the number of atoms, volume and temperature are held constant. This means the pressure is allowed to fluctuate in an NVT ensemble.

3.3.2.5 Langevin Dynamics

Langevin Dynamics is used to keep a constant temperature in a MD simulation. This is done by approximating the effect the (solvent) environment has on the system through a friction term:[48]

$$m\vec{a} = \vec{F} - \gamma\vec{v} + R(t) \quad \textbf{[44]}$$

where $\vec{F}$ is the force on the individual atom in the simulation obtained from equation [37], m is the atomic mass of the atom, $\vec{a}$ is the acceleration of the atom, γ is the friction coefficient, $\vec{v}$ is the velocity of the atom, and $R(t)$ is the Gaussian random force. The additional terms in equation [44] as compared to equation [38] causes the first derivative of position $\vec{r}$ to no longer be velocity $\vec{v}$, and the derivative of velocity is no longer acceleration $\vec{a}$. The Brünger-Brooks-Karplus integrator (equation [45]) is used instead of equation [40] in the Velocity Verlet algorithm.

$$\vec{r}_{(t+\delta t)} = \left(1 + \frac{\gamma(\delta t)}{2}\right)^{-1} \times \left(2\vec{r}_t - \vec{r}_{(t-\delta t)} + \frac{(\delta t)^2}{m}\left(\vec{F} + R(t)\right) + \frac{\gamma(\delta t)}{2}\vec{r}_{(t-\delta t)}\right) \quad \textbf{[45]}$$

3.3.2.6 Berendsen Pressure

Berendsen Pressure bath coupling is an algorithm used to maintain constant pressure in a MD simulation. The pressure of the system affects the size of the simulation box at equilibrium; the more pressure applied, the smaller the simulation box. The algorithm weakly couples the system to an external pressure bath. To implement the constant pressure control, a scaling factor μ is added to some variables in Newton's equations:[55]

$$\mu = \left(1 - \frac{(\delta t)(P_0 - P)}{\tau_P}\right)^{1/3} \quad \textbf{[46]}$$

where μ is the scaling factor, δt is the time step, P_0 is the pressure of the external pressure bath, P is the pressure of the system, and τ_P is the time constant for coupling.

3.3.3 Replica Exchange Molecular Dynamics

Replica Exchange Molecular Dynamics (REMD) is a method used to enhance conformational sampling of the molecular system under investigation.[56–58] REMD uses a combination of MD and Monte Carlo (MC) steps in the simulation. REMD uses several copies (replicas) of the molecular system, each at a different temperature, using MD simulations run in parallel. The MD simulations at different temperatures are run in parallel (isolated from each other) for a predetermined number of time steps after which an MC step is run. The goal of the MC step is to attempt an exchange or swap of temperatures between pairs of MD replicas (Figure 10). Consider simulation i with molecular coordinates $\vec{R}_i$ at temperature T_1 and simulation j with molecular coordinates $\vec{R}_j$ at temperature T_2. The exchange step attempts$(\vec{R}_i, T_1; \vec{R}_j, T_2) \rightarrow (\vec{R}_i, T_2; \vec{R}_j, T_1)$. If the exchange step is successful, the deterministic Newton's laws used to compute atomic positions (trajectories) are broken for both simulations i and j. If the exchange step is unsuccessful, both simulations continue for the next interval of MD steps and the trajectories are unbroken.

The occasional exchange between replicas enhances conformational sampling by allowing conformations more accessible at high temperatures to be introduced into a low temperature simulation. This allows REMD simulations to overcome the multiple minimum problem better than classical MD simulations. This reduces the computational time and produces conformations which might not have been explored in the MD simulations alone. By switching

the higher energy conformations to the lower temperature simulations, the barriers between local minima in the potential energy function can be overcome.[57]

Figure 10 shows five different replicas exchanging five different temperatures over a 50000 time step period. Only adjacent replicas are allowed to exchange temperatures. Over a period of 33000 time steps, the temperature T_1moved from replica 1 all the way to replica 5. Over time steps 0 to 12000, none of the attempted exchanges between T_1 and T_2 were accepted, as shown by the horizontal line. Over the time 30000 and 33000 all the attempted exchanges were accepted for T_1, as shown by the continuous diagonal line.

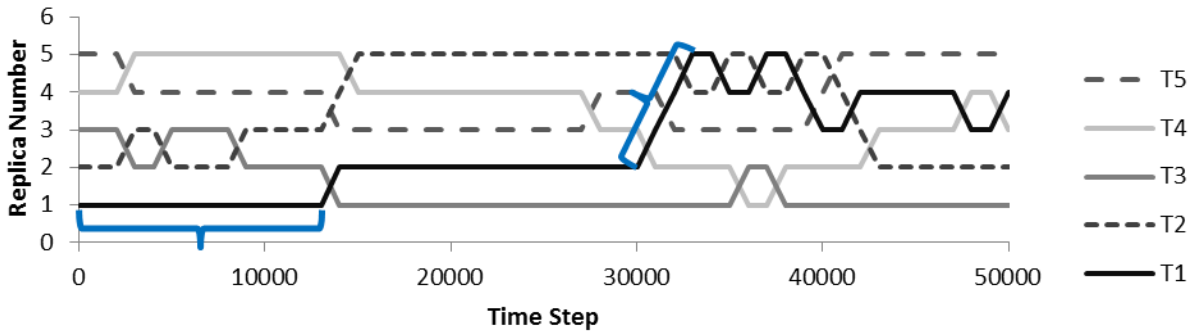


Figure 10: The replica number versus time step over a period of 50000 time steps. Only adjacent replicas can exchange temperatures; exchanges are attempted every 1000 time steps.

The MC step is run after a set number of time steps in an REMD simulation (exchange attempt).[59] The number of time steps is usually chosen to allow the solvent to equilibrate at the new temperature.[59,60] Only replicas which are adjacent in temperature have the ability to exchange their temperatures.[56,57] This is done by utilizing an acceptance ratio, $\pi(X_2 \rightarrow X_1)$, derived from the detailed balance equation:[57,58]

$$\pi(X_2 \rightarrow X_1) = \begin{Bmatrix} \Delta\beta * \Delta U & \leq 0 \\ \exp(-\Delta\beta * \Delta U) & > rand \end{Bmatrix} \quad \textbf{[47]}$$

where $$\Delta\beta = \frac{1}{k_B T_1} - \frac{1}{k_B T_2}$$ **[48]**

and $$\Delta U = U_2 - U_1$$ **[49]**

where $\Delta\beta$ and ΔU are calculated from equations [48] and [49] respectfully. The $T_1$is the lower temperature, T_2 is the higher temperature and k_B is the Boltzmann's constant. The potential energies at the two different temperatures are $U_1$and U_2, and $rand$ is a uniform random number between zero and one, which is supplied by the program. When $(-\Delta\beta*\Delta U)$ is a large negative number, the $\exp(-\Delta\beta*\Delta U)$ goes to zero and when $(-\Delta\beta*\Delta U)$ is a small negative number the $\exp(-\Delta\beta*\Delta U)$ goes to one. The condition to accept or reject the switch is given in equation [47]: if $\Delta\beta*\Delta U$ is less than zero or $\exp(-\Delta\beta*\Delta U)$ is greater than the random number (between 0 and 1), replica exchange will take place. The second condition implements a measure of chance, where a non-frequent conformation can be accepted. When $(-\Delta\beta*\Delta U)$ is a small negative number ΔU is small, this means the potential energy difference is small. The probability of the exchange being accepted is higher when the function goes to one than when the function goes to zero. If the potential energy overlaps there is still a possibility the exchange will be rejected. If a conformation occurs more frequently, the solution to the expression $\exp(-\Delta\beta*\Delta U)$ will be closer to 1. However, because of the random number conformations across large energy barriers can be explored.

The acceptance ratio for replica exchanges is calculated to ensure that enhanced sampling takes place. An appropriate exchange ratio is between 0.2 and 0.5.[60] For the exchange ratio to be appropriate, there must be sufficient overlap in the potential energy distributions for two adjacent temperatures (Figure 11). If the distributions do not overlap, there is zero probability exchanges will take place. The difference in energy ΔU in equation [47] will be too large and the exchange

will be rejected. If the potential energy distributions overlap too much, the exchanges will always be accepted. The difference in energy ΔU in equation [47] will be very small and the exchange will always be accepted. In this case, the conformational distributions explored at these two temperatures are almost identical and no enhanced sampling has occurred.

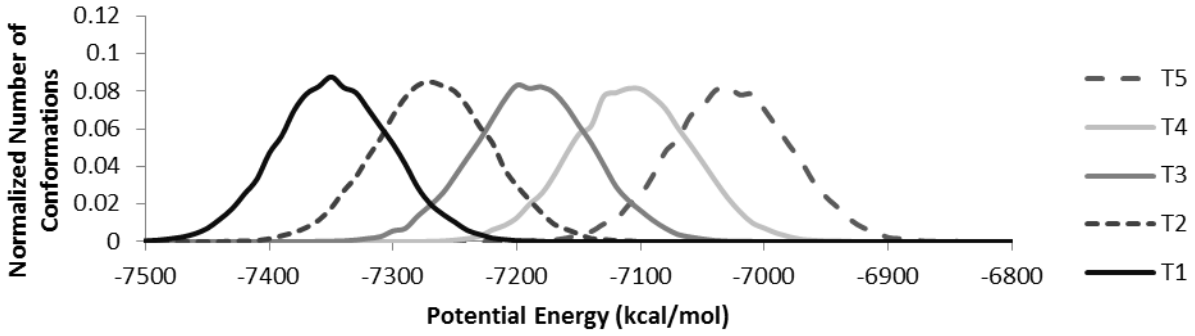


Figure 11: Potential energy histogram of the five replicas organised by target temperature. Each replica has an area under the curve as 1.

The temperature range and number of replicas are designated by the user. These are tested until the exchange ratio gives an appropriate value. In this work, the temperature range was kept the same and the number of replicas were changed to obtain the desired exchange ratio.

REMD most conveniently uses an NVT ensemble and not an NPT ensemble. Equation [47] is only valid in an NVT ensemble. This particular acceptance condition is derived for replicas at constant density (N/V).[15] To perform an REMD simulation for the NPT ensemble, a different exchange condition would be needed.[61]

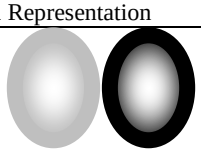
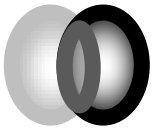
3.4 Conformational Space Explored by Molecular Simulations

It is nontrivial to quantitatively compare the extent of conformational space explored by molecular simulations. One way of comparing conformations explored during different simulations is by plotting the potential energy of the system as a function of time. If the span of

potential energies is different, the conformations explored by the simulations are not similar. However, if the potential energies are the same, the conformations explored might be identical or their three-dimensional structures happen to produce similar potential energies.

Table 5 uses Venn Diagrams to explain the outcomes that can happen when comparing conformational distributions produced by different simulations of the same system. Each oval represents the conformational diversity explored by a single MD simulation. If two independent simulations of the same system explored completely different space in the conformational distributions, the ovals would have no overlap. This result is not ideal for two independent simulations of the same system. If the ovals completely overlap, the independent simulations of the same system starting from two different positions have sampled the same conformational space. If the simulations only sampled a portion of each other's conformational space, this means neither simulation was run long enough to cover the complete conformational distribution. The ovals would only partially overlap. The last case is when one simulation covers a subset of another simulation. This is an example of a system exploring conformations of a deep energy well where the simulation could not get out of it. A REMD simulation is an enhanced sampling technique and could move the system's conformation out of a deep energy wells where an MD simulation could not.

Table 5: Pictorial representation of comparing protein conformational ensembles from different simulations by Venn Diagrams

Pictorial Representation	Explanation
	No overlap of conformational ensembles: the simulations **do not explore the same conformational space**
	Complete overlap of conformational ensembles: the simulations **explore the same conformational space**
	Partial overlap of conformational ensembles: the simulations **partially explore each other's conformational space**
	Complete overlap in the conformational ensemble for one but not the other: one simulation explores a **subset of the second simulation's conformational space**

4 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a way of reducing the dimensionality of a high-dimensional dataset.[62] PCA generates and then diagonalizes the variance-covariance matrix of the dataset and orders the resulting eigenvectors in decreasing contribution to the total sample variance. These values in decreasing order are known as eigenvalues. A new set of axes is defined so each eigenvector is uncorrelated or orthogonal to the rest.[63] The principal component (PC) eigenvectors are linear combinations of the original (correlated) variables describing the data.[64] The maximum number of PC is the same as the original dimensions describing the dataset. The reduction in dimensionality occurs because usually a small subset of the PCs with the largest eigenvalues describes the majority of the sample variance.

4.1 Dihedral Principle Component Analysis (dPCA)

The program *carma* was used in this work to compute the dihedral principle component analysis (dPCA).[65,66] The program generates the transformed dihedral angles from the ψ angles

of the first amino acid to the φ angle of the last amino acid of a conformational dataset describing a peptide or protein.[10,65] The program outputs eigenvalues and eigenvectors for all dihedral dimensions. The eigenvectors are orthogonal to each other and the sum of squared coefficients of the eigenvectors over all dimensions equals unity.[10] The eigenvalues are ordered in decreasing contribution to the total sample variance.[65] These values can be used to determine which PCs should be included in subsequent analysis, as describing most of the conformational variability. In this work the PCs retained explained 80% of the variance in the dataset.

5 Circular Statistics of Angular Variables

The datasets used to train the SOMs and the datasets used in the dPCA are generated from dihedral angles that were transformed into metric coordinate space (unit circle),[10,67] to account for the circular statistics of angular variables. This means the programs will recognize the periodic nature of dihedral angles, where 180° is equivalent to -180°. Each dihedral angle is converted into its components x and y: [10,67]

$$\varphi \rightarrow \begin{Bmatrix} x = cos\varphi \\ y = sin\varphi \end{Bmatrix} \quad \textbf{[50]}$$

where φ is one of the dihedral angles (either φ or ψ). This means the dimensionality of the dataset is doubled to account for the transformation.

6 Goals

The aim of this work is to devise a protocol to cluster peptide or protein conformations generated by MD or REMD computer simulations onto an SOM. This protocol must include an evaluation to determine if SOMs can be reproducible and to accommodate any size (number of

amino acids) of protein. Two datasets are used to help develop the SOM protocol: the flexible penta-peptide Met-enkephalin (MET) and a flexible 84-amino acid protein (hPTH).

The penta-peptide MET was used to determine if C_v is an appropriate way to compare independently trained SOMs from the same dataset and to determine the reproducibility of the data partitioning. The protocol for SOM training using conformations of the penta-peptide MET protocol was applied further to clustering molecular conformations of a larger flexible protein hPTH.

7 Met-Enkephalin (MET)

Enkephalins are endogenous penta-peptides that show opiate activity similar to morphine.[68] In 1975, two forms of enkephalins were isolated from pig's brain extracts: Met-enkephalin (MET; YGGFM) and Leu-enkephalin (YGGFL).[69] Of these two enkephalins, MET is the most active.[70] MET is found in the central nervous system and the gastrointestinal tract.[71] It preferentially binds to the δ-opioid receptor, but it also interacts with the μ-opioid receptor and the opioid growth factor receptor.[59,71] Since MET has important physiological roles, this neuropeptide has been extensively studied both experimentally and by computer simulations.[72,73] Due to the extensive research on MET, it is often used as a bench mark model for testing new simulation methods because of its small size and flexibility.[74,75] Many millions of conformations of MET can be generated by MD simulations in a short period of user-time.

In solution, MET assumes a large number of conformations resulting in a broad conformational distribution.[59,70–72,74] The flexibility of MET explains the interaction with multiple opioid receptors. Furthermore, the flexibility of MET has made three-dimensional

structure determination difficult, meaning that no unique structure in solution has been determined experimentally by NMR.[71]

7.1 Simulations with MET

Many reports of simulations of MET are in the literature, two of which are discussed here.[59,72,74,75] More varied conformations are produced by MET simulated in solvent than in a vacuum; fewer intra-peptide hydrogen bonds are formed in the more polar environment of aqueous solvent, which increases peptide flexibility.[72] Sanbonmatsu and García studied the conformations of the protected N- and C- terminal MET (by N-acetylamide and N-amide groups, respectively) in explicit water (TIP3P),[59] and found that MET has four predominant structures, two helical and two non-helical in shape. They ran REMD simulations (NVT ensemble) with 16 replicas, where the temperatures ranged from 275-419 K. The simulations were run for a total of 32 ns, then the conformations obtained at 275 K were clustered by their first two principal components (PC). They determined that REMD explored more conformational space than classical MD simulations. They did this by generating a classical MD simulation with an NVT ensemble for the same number of timesteps and performed the same conformational analysis. The results showed that one conformation (a helical conformation) appeared predominantly over the other three at 275 K.

Malevanets and Wodak's research on a new sampling technique, multiple replica repulsion, generated the broadest conformational distribution of the protected N- and C- terminal MET (same as Sanbonmatsu and García).[74] The simulation was an NVT ensemble with 1000 TIP3P waters, run for 32 ns at a constant temperature of 300 K. Malevanets and Wodak compared the five amino acid Ramachandran plots of their multiple replica repulsion to those of Sanbonmatsu and García. Malevanets and Wodak found their work with the new sampling

technique covered a broader conformational range than did Sanbonmatsu and García's REMD simulations.

7.2 Methods and Materials for MET

7.2.1 Preparation

Sanbonmatsu and García's research on the conformations of Met-Enkaphalin in explicit water[59] formed the basis for our simulation protocol. They ran a replica exchange molecular dynamics simulation for 32 ns. The starting MET conformations of the NMR-solved structure were taken from the protein DataBank[76] structure in 1PLX: models 1 (sample 1; Figure 12B) and 66 (sample 2; Figure 12C).[71] These two different structures are visually similar to the non-helical structures found by Sanbonmatsu and García. These structures were modified using the VMD[77] *molefacture* plugin; the C-terminus was amidated and the N-terminus was acetylated (Figure 12A), resulting in a neutral charge peptide and thereby significant intramolecular electrostatic interactions between end termini were eliminated.

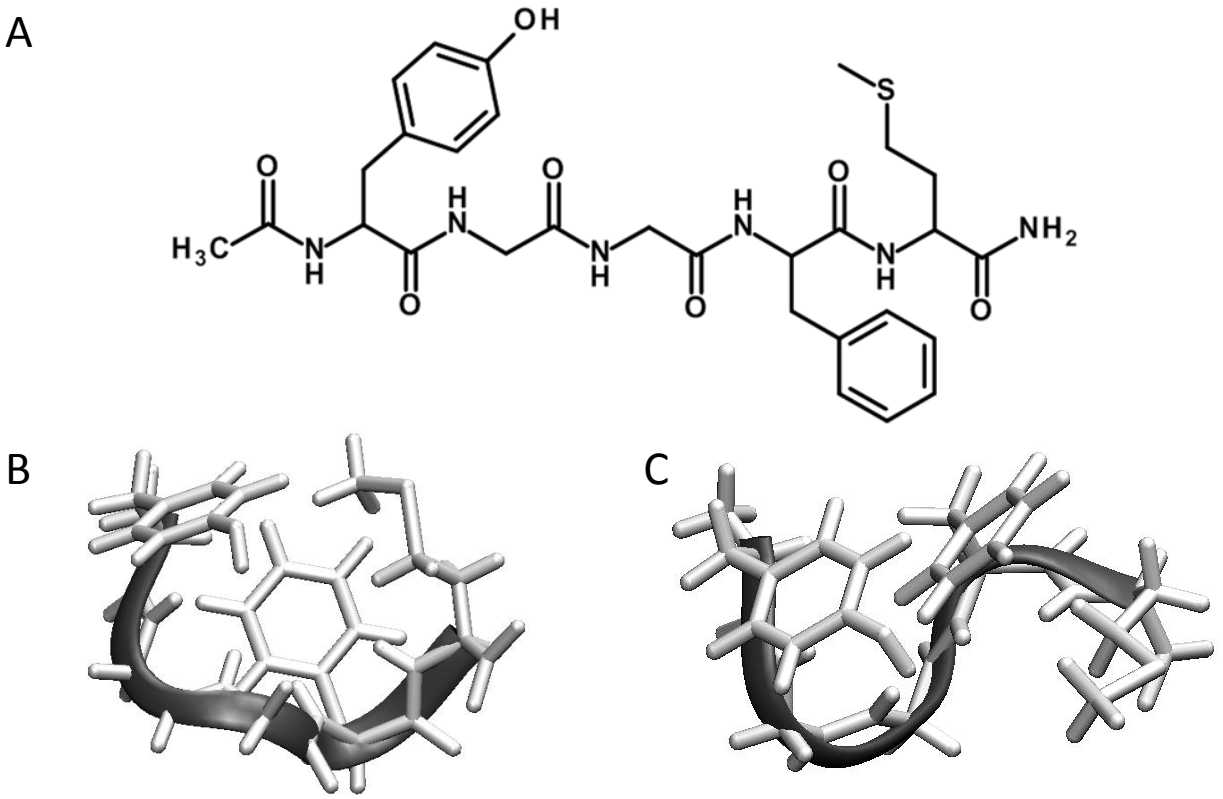


Figure 12: Images of the protected C and N terminus of MET (YGGFM). N-terminus protected by acetylation and C-terminus protected by amidation. (A) 2D image produced using Accelrys Draw 4.1, (B) Sample 1 from model 1 from 1PLX, and (C) Sample 2 from model 66 from 1PLX

7.2.2 Vacuum MD Simulations

Two all-atom molecular dynamics (MD) simulations were performed on the protected penta-peptide MET without explicit solvent using NAMD 2.7[48] on SHARCNET[78] in an NVE ensemble using the CHARMM c31b1 force field.[46] One simulation, vac/MD1, which used model 1 of 1PLX as the starting conformation was run by Mark Cooper Gienow[79] and the second simulation, vac/MD2, used model 66 was run by Michelle A. Eisner. Two simulations were run to compare if the simulations explored the same conformational space. All pairwise van der Waals and electrostatic interactions were included. The dielectric constant ε was set to 1. Covalently bonded hydrogens were constrained to their equilibrium bond distances using the

SHAKE algorithm.[51] The potential energy of the initial conformation was minimized for 1000 conjugate gradient steps in order to reduce unfavourable interactions. The Velocity Verlet algorithm was used to perform the MD simulation on the system. An integration time step of 2 fs was used. The temperature was increased from 0 K to 300 K by intervals of 25 degrees every 1000 time steps. The temperature was then held at 300 K using Langevin dynamics.[48] The systems were equilibrated for 2 ps. The simulations were run for 64 ns and the atomic coordinates were written to trajectory files every 20 ps for a total of 3200 conformations, which were used for subsequent analysis. Ramachandran plots were generated for the five amino acids to assess the range of conformational space explored throughout both simulations.[59]

7.2.3 Solvated MD Simulations

Two all-atom molecular dynamics (MD) simulations were performed on the protected MET with explicit solvent using NAMD 2.7[48] on SHARCNET[78] in an NPT ensemble using the CHARMM c31b1 force field.[46] One simulation, TIP3P/MD1, was run by Mark Cooper Gienow[79] and Michelle A. Eisner and started from model 1 of 1PLX and the second simulation, TIP3P/MD2, was run by Michelle A. Eisner and started from model 66. The peptide was solvated by 826 TIP3P[49] water molecules contained in a cubic simulation cell with an edge length of 30 Å. The potential energy of the system was calculated using periodic boundary conditions. The cutoff distance for the non-bonded terms was 12 Å and the switching distance was 8 Å. The pair-list distance was generated between non-bonded atoms under 13.5 Å and was generated twice per cycle, where the cycle was 20 time steps. Covalently bonded hydrogens were constrained to their equilibrium bond distances using the SHAKE algorithm.[51] The potential energy of the system of MET plus 826 TIP3P molecules was minimized for 1000 conjugate gradient steps in order to reduce the unfavourable interactions due to atom overlap between the

TIP3P water and the peptide. Langevin dynamics[48] and Berendsen pressure bath coupling[55] were used to maintain a constant temperature at 300 K and a constant pressure of 1 atm. The Velocity Verlet algorithm was used to perform the MD simulation on the system. An integration time step of 2 fs was used. The system was equilibrated for 2 ps. The simulation was run for 64 ns and the coordinates were written to trajectory files every 20 ps for a total of 3200 conformations, which were used for subsequent analysis.

7.2.4 Solvated REMD Simulations

Using the coordinates of the last frames from each of the solvated MD simulations as starting conformations, two different REMD simulations, TIP3P/REMD1 and TIP3P/REMD2, were run. The REMD simulations were carried out using NAMD 2.9[48] on SHARCNET[78] in an NVT ensemble using the CHARMM c31b1 force field.[46] Both REMD simulations had 16 replicas running in parallel at different temperatures: 275, 282.83, 290.88, 299.16, 307.68, 316.44, 325.45, 334.72, 344.25, 354.05, 364.13, 374.49, 385.16, 396.12, 407.4, 419 K. These temperatures correspond to the temperatures used in Sanbonmatsu and García's research.[48] Exchanges between adjacent temperature replicas were attempted every 2.0 ps and coordinates were written to trajectory files every 20 ps. The average exchange ratio for both TIP3P/REMD1 and TIP3P/REMD2 was 19.5%, which is similar to the 22.5% reported by Sanbonmatsu and Garcia.[59] An energy histogram was created to show that sufficient overlap of the potential energy (Figure 13) existed to ensure an adequate exchange ratio. The REMD simulations were carried out for 68 ns using an integration step of 2 fs, producing a total of 3400 conformations per replica. The conformations at 299.16 K were analysed since that temperature was the closest to the original temperature in the MD simulations TIP3P/MD1 and TIP3P/MD2 at 300K.

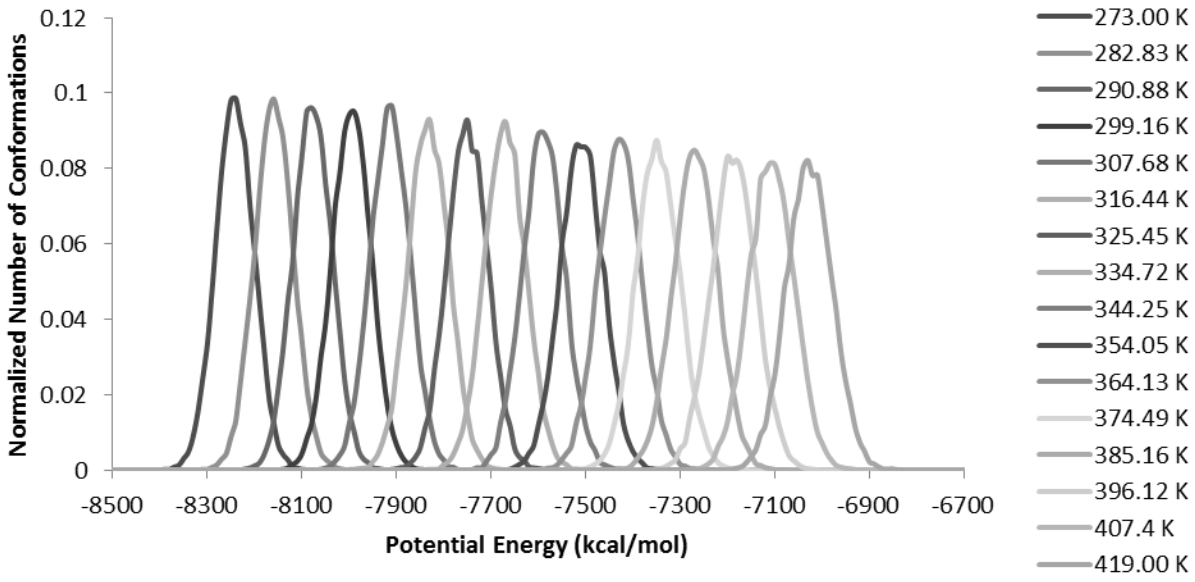


Figure 13: Energy histogram of the replicas in TIP3P/REMD1 organised by target temperature. Each replica has an area under the curve of 1.

7.2.5 SOM Analysis on MET

The backbone dihedral angles ϕ and ψ were extracted from the conformations of MET produced from the MD simulations (3200 conformations) and the REMD simulations (3400 conformations). The dihedral angles were transformed into metric coordinate space,[10,67] to account for the circular statistics of angular variables, producing a 20-dimensional dataset (5 amino acids × 2 angles × 2 components = 20 dimensions). The names of the datasets are in Table 6. The datasets were then used to train the different SOMs.

Table 6: Names of the 20-dimensional datasets generated from the different simulations of MET

Name of Dataset	Sample	Environment	Simulation Type	Number of Conformations
vac/MD1	1	vacuum	Classical MD	3200
vac/MD2	2	vacuum	Classical MD	3200
TIP3P/MD1	1	Solvated (TIP3P)	Classical MD	3200
TIP3P/MD2	2	Solvated (TIP3P)	Classical MD	3200
TIP3P/REMD1	1	Solvated (TIP3P)	REMD	3400
TIP3P/REMD2	2	Solvated (TIP3P)	REMD	3400

The different SOMs were produced using a number of different parameters. Table 7 is a list of parameters associated with generating all the SOMs described in this chapter. The parameters are labeled 7-a to 7-zz where the number is the heading number and the letter(s) is associated with a particular protocol. Table 7 is presented in order of appearance in the chapter. When discussing untrained maps, unimplemented functions are represented by (---).

Table 7: Parameters used to make bordered and toroidal SOMs of the MET datasets

Para	Dataset	Program	$\eta \times \eta$	Nodal Geom[a]	Initial[b]	Train[c]	η_τ[c]	$\alpha(\tau)$[d]	T[c]	# of SOMs
7-a	TIP3P/MD1	C++	5×5	rect	maxmin	Seq	[9]	[14] [iii]	100	500
7-b	TIP3P/MD1	C++	5×5	rect	rand	Seq	[9]	[14] [iii]	100	500
7-c	TIP3P/MD1	C++	5×5	rect	rvec	Seq	[9]	[14] [iii]	100	500
7-d	TIP3P/MD1	C++	5×5	rect	maxmin	---	---	---	0	500
7-e	TIP3P/MD1	MATLAB	5×5	rect	maxmin	---	---	---	0	100
7-f	TIP3P/MD1	MATLAB	6×6	hexa	maxmin	---	---	---	0	100
7-g	TIP3P/MD1	C++	7×7	rect	maxmin	---	---	---	0	500
7-h	TIP3P/MD1	MATLAB	7×7	rect	maxmin	---	---	---	0	100
7-i	TIP3P/MD1	C++	10×10	rect	maxmin	---	---	---	0	500
7-j	TIP3P/MD1	MATLAB	10×10	rect	maxmin	---	---	---	0	100
7-k	TIP3P/MD1	MATLAB	10×10	hexa	maxmin	---	---	---	0	100
7-l	TIP3P/MD2	C++	10×10	rect	maxmin	---	---	---	0	100
7-m	TIP3P/MD2	MATLAB	10×10	rect	maxmin	---	---	---	0	100
7-n	TIP3P/MD2	MATLAB	10×10	hexa	maxmin	---	---	---	0	100
7-o	TIP3P/REMD1	C++	10×10	rect	maxmin	---	---	---	0	100
7-p	TIP3P/REMD1	MATLAB	10×10	rect	maxmin	---	---	---	0	100
7-q	TIP3P/REMD1	MATLAB	10×10	hexa	maxmin	---	---	---	0	100
7-r	TIP3P/REMD2	C++	10×10	rect	maxmin	---	---	---	0	100
7-s	TIP3P/REMD2	MATLAB	10×10	rect	maxmin	---	---	---	0	100
7-t	TIP3P/REMD2	MATLAB	10×10	hexa	maxmin	---	---	---	0	100
7-u	vac/MD1	C++	5×5	rect	maxmin	Seq	[8]	[13]	100	3
7-v	TIP3P/MD1	C++	5×5	rect	maxmin	Seq	[8]	[13]	100	3
7-w	TIP3P/MD1	MATLAB	10×10	rect	maxmin	Seq	[10]	[16] [ii]	100	100
7-x	TIP3P/MD1	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	100	100
7-y	TIP3P/MD2	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	100	100
7-z	TIP3P/MD2	MATLAB	10×10	rect	maxmin	Seq	[10]	[16] [ii]	100	100
7-aa	TIP3P/REMD1	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	100	100
7-bb	TIP3P/REMD1	MATLAB	10×10	rect	maxmin	Seq	[10]	[16] [ii]	100	100
7-cc	TIP3P/REMD2	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	100	100
7-dd	TIP3P/REMD2	MATLAB	10×10	rect	maxmin	Seq	[10]	[16] [ii]	100	100
7-ee	TIP3P/MD1	C++	7×7	rect	maxmin	Seq	[9]	[14] [iii]	100	500
7-ff	TIP3P/MD1	C++	10×10	rect	maxmin	Seq	[9]	[14] [iii]	100	500
7-gg	TIP3P/MD1	MATLAB	5×5	rect	maxmin	Seq	[10]	[15] [i]	100	100
7-hh	TIP3P/MD1	MATLAB	7×7	rect	maxmin	Seq	[10]	[15] [i]	100	100
7-ii	TIP3P/MD1	MATLAB	10×10	rect	maxmin	Seq	[10]	[15] [i]	100	100
7-jj	TIP3P/MD1	MATLAB	10×10	hexa	maxmin	Seq	[10]	[15] [i]	100	100
7-kk	TIP3P/MD1	MATLAB	5×5	rect	maxmin	Batch	---	---	100	100
7-ll	TIP3P/MD1	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [i]	100	100
7-mm	TIP3P/MD1	MATLAB	10×10	hexa	maxmin	Seq	[10]	[17] [i]	100	100
7-nn	TIP3P/MD1	MATLAB	10×10	hexa	7-jj	Seq	[10]	[16] [ii]	100	100
7-oo	TIP3P/MD1	MATLAB	10×10	hexa	7-ll	Seq	[10]	[16] [ii]	100	100
7-pp	TIP3P/MD1	MATLAB	10×10	hexa	7-mm	Seq	[10]	[16] [ii]	100	100
7-qq	TIP3P/MD1	MATLAB	10×10	hexa	7-jj	Seq	[10]	[17] [ii]	100	100
7-rr	TIP3P/MD1	MATLAB	10×10	hexa	7-ll	Seq	[10]	[17] [ii]	100	100
7-ss	TIP3P/MD1	MATLAB	10×10	hexa	7-mm	Seq	[10]	[17] [ii]	100	100
7-tt	TIP3P/MD1	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	500	100
7-uu	TIP3P/MD1	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	1000	100
7-vv	TIP3P/MD1	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	5000	100
7-ww	TIP3P/MD1	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	10000	100
7-xx	TIP3P/MD2	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	5000	100
7-yy	TIP3P/REMD1	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	5000	100
7-zz	TIP3P/REMD2	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [ii]	5000	100

[a] description can be found in section 2.1
[b] description can be found in section 2.3
[c] description can be found in section 2.4; square brackets refer to equation numbers
[d] description can be found in section 2.4.2.1; square brackets refer to equation numbers
--- function which is not relevant for untrained map

[i] $\alpha_0 = 0.5$
[ii] $\alpha_0 = 0.05$
[iii] $\alpha_0 = 0.1$

7.3 Results and Discussion: Clustering Conformations of MET

7.3.1 Initialization

There are three different types of initializations of the SOMs which are possible using the C++ program: maxmin, rand, and rvec (Section 2.3). Of these three programs, only maxmin is available in the MATLAB program. Ideally, the clustering produced through the training procedure should be independent of the map initialization. The map initialization was tested to determine if the trained maps produced from the different initializations produced the same results. Figure 14 plots the C_v distributions for 500 5×5 SOMs of the TIP3P/MD1 dataset which were initialized by different algorithms (Table 6: 7-a - 7-d). These algorithms are identical in training these maps; the only difference is from the initialization. Two boundary conditions were used to train these maps; the dashed lines are toroidal maps while the solid are bordered maps. The C_v distribution is not dependent on the map initialization. When comparing the toroidal and bordered trained maps by C_v distributions, the C_v values for toroidal maps are shifted to higher values than for bordered maps. These boundary conditions cluster the data differently. This will be discussed further in Section 7.3.3.

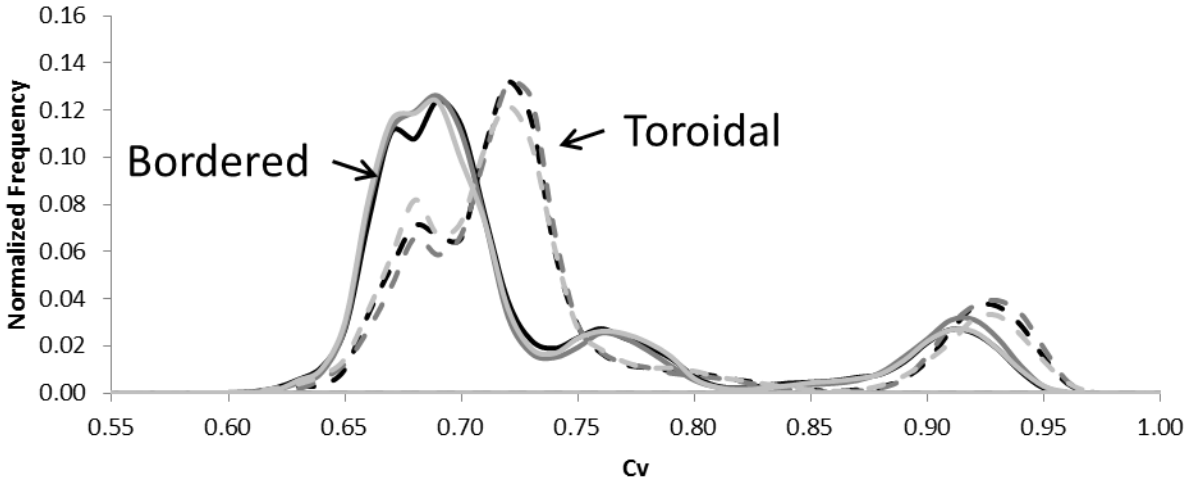


Figure 14: The C_v distribution for the TIP3P/MD1 5×5 rectangular SOMs initialized by three different algorithms (maxmin, rand, and rvec) but trained by the exp ↓NEIGH in the C++ program. There are 500 SOMs for each map type. ■ is the maxmin initialization (Table 7:7-a), ■ is the rand initalization (Table 7:7-b), and ■ is the rvec initialization (Table 7:7-c). Solid lines are for bordered boundaries; dotted lines are for toroidal boundaries.

It is expected that given the same mapsize η and initialization method, that the two SOM programs would generate similar initial random partitionings of the data. The effects of different mapsizes η, geometries (rectangular and hexagonal), and programs (C++ and MATLAB) were examined by comparing C_v distributions for untrained maps (Figure 15). The initialization maxmin is chosen for all cases. Note that the C_v values for untrained maps (Figure 15) are much lower than for the trained SOMs (Figure 14). The initial random placement of data on the SOM produces many unlike partitionings. The subsequent SOM clustering does not result in identical maps but they all bear data partitionings more similar at the end of the training. The rectangular geometry for the C++ program and the rectangular geometry for the MATLAB program produced similar C_v distributions for 5×5, 7×7, and 10×10 untrained maps (Figure 15). As the mapsize increases, the C_v histogram shifts to the right and the histograms narrow. The initial random data partitioning is independent of geometry: the C_v values for 10×10 hexagonal untrained maps overlap with those for the 10×10 rectangular untrained maps.

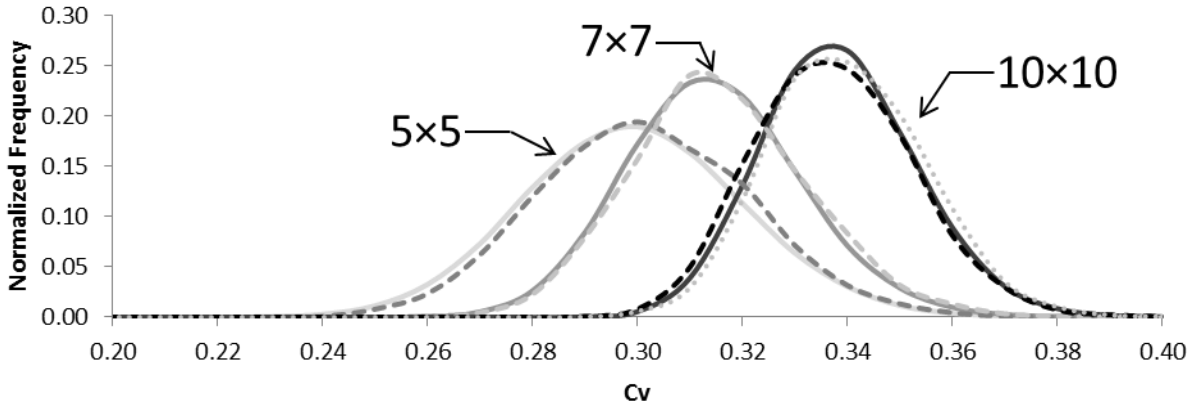


Figure 15: The C_v distribution for the same dataset and different size maps of the untrained SOMs. The SOMs were trained by the maxmin initialization algorithm and by the TIP3P/MD1 dataset: (—) untrained rectangular 5×5 from the C++ program (Table 7:7-d), (– –) untrained rectangular 5×5 from the MATLAB program (Table 7:7-e), (…) untrained hexagonal 6×6 from the MATLAB program (Table 7:7-f), (—) untrained rectangular 7×7 from the C++ program (Table 7:7-g), (– –) untrained rectangular 7×7 from the MATLAB program (Table 7:7-h), (—) untrained rectangular 10×10 from the C++ program (Table 7:7-i), (– –) untrained rectangular 10×10 from the MATLAB program (Table 7:7-j), and (…) untrained hexagonal 10×10 from the MATLAB program (Table 7:7-k).

Next, the different datasets were compared by the C_v distributions for untrained SOMs using maxmin initialization (Figure 16). The untrained SOMs that were initialized by maxmin and using datasets generated from REMD simulations have lower C_v values than those from the datasets generated by classical MD simulations. The separation among the C_v distributions is a consequence of the different diversities within the conformational distributions and the use of maxmin initialization. Instead, if the rand initialization was used, identical narrow C_v distributions are seen for both REMD and MD datasets centered at ~0.18 (not shown). The rand initialization is dependent only on the total number of conformations and the number of groups into which they are divided, whereas maxmin initialization is dependent on the vectors describing the conformations. It is known that REMD is a more efficient sampling technique than classical MD simulations. If the REMD datasets are more conformationally diverse than the classical MD datasets, the C_v distribution for untrained SOMs generated for REMD datasets will be shifted towards lower values than those obtained for the classical MD simulations. This is, the range between the minimum and maximum values for each of the conformational dimensions

will be broader for a more diverse sample, resulting in more diverse untrained SOMs bearing less similarity to each other. Similarly TIP3P/REMD1 appears to cover more conformational space than TIP3P/REMD2 as well as TIP3P/MD1 covers more conformational space than TIP3P/MD2. This is due to the shift towards lower C_v values of the untrained SOMs.

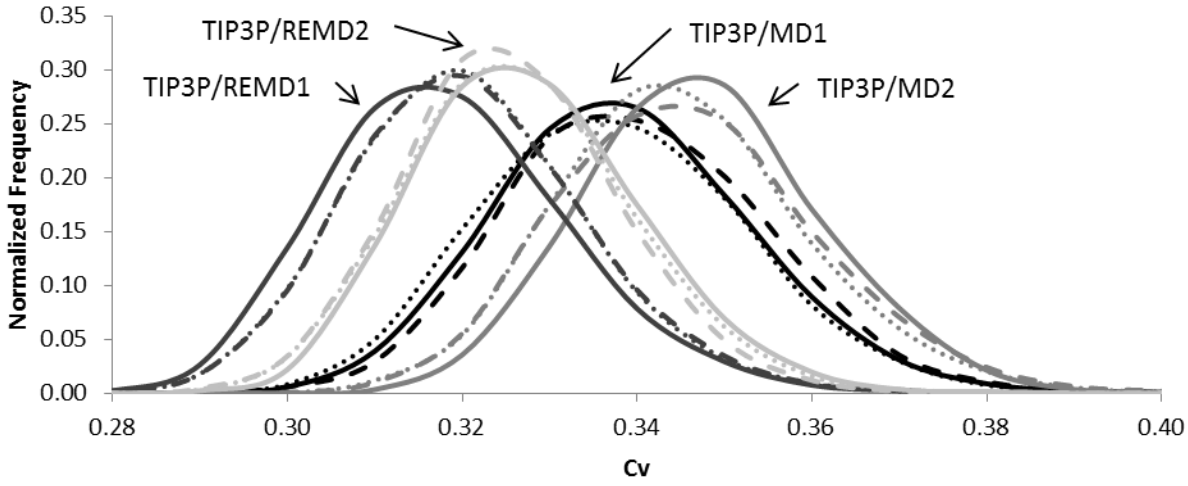


Figure 16: The C_v distribution of untrained 10×10 SOMs generated by four different datasets. The colours represent the datasets used to initialize the SOMs: ■ is the TIP3P/MD1 dataset (Table 7:7-i, 7-j, and 7-k), ■ is the TIP3P/MD2 dataset (Table 7:7-l, 7-m, and 7-n), ■ is the TIP3P/REMD1 dataset (Table 7:7-o, 7-p, and 7-q), and ■ is the TIP3P/REMD2 dataset (Table 7:7-r, 7-s, and 7-t). The lines represent the geometry and the program used to initialize the SOMs: (——) are rectangular SOMs the C++ program, (.....) are rectangular SOMs the MATLAB program, and (– –) are hexagonal SOMs the MATLAB program. 100 SOMs were initialized except TIP3P/MD1 dataset, with the C++ program where 500 SOMs were initialized.

7.3.2 Conformational Space of MET Explored by Molecular Simulations

Two methods used in this work to compare conformational distributions for the peptide MET are Ramachandran plots and using trained SOMs as a supervised classification method. When simulating the peptide MET, the ideal situation is either having complete overlap of the conformational ensembles produced from independent simulations, or having an initial prediction verified by the conformational overlay of the ensembles. Table 6 shows the six simulations that were all run from two different starting conformations. It can be seen from visualization of the trajectories via VMD that the simulations run without solvent cover a smaller

conformational diversity than the simulations run in explicit solvent. This means when conformations of vac/MD1 or vac/MD2 are classified using an SOM trained using an explicitly solvated dataset as a supervised classification algorithm, the simulations run in a vacuum should populate a subset of the nodes of that SOM. This is congruent with the results of Sanbonmatsu and García.[59] Complete overlap should happen when REMD simulations are compared to each other, if they both independently explored the entire Mat conformational space. When REMD simulations are compared to MD simulations, REMD simulations should explore a broader conformational distribution since it is a more efficient sampling technique. MD simulations should explore a subset of REMD simulations.

7.3.2.1 Ramachandran Plots

Ramachandran plots were generated on a per residue basis for all the conformational ensembles in Table 6. These plots depict the ϕ and ψ backbone dihedral angles of all the residues in the peptide. Ramachandran plots are a way to cluster proteins into like conformations of the backbone dihedral angles. There are limitations: there is no way to link the different residues together (i.e. identify correlated values of φ and ψ), nor is there a way to definitively locate clusters since several conformational clusters could appear on top of each other and visually look like one cluster.

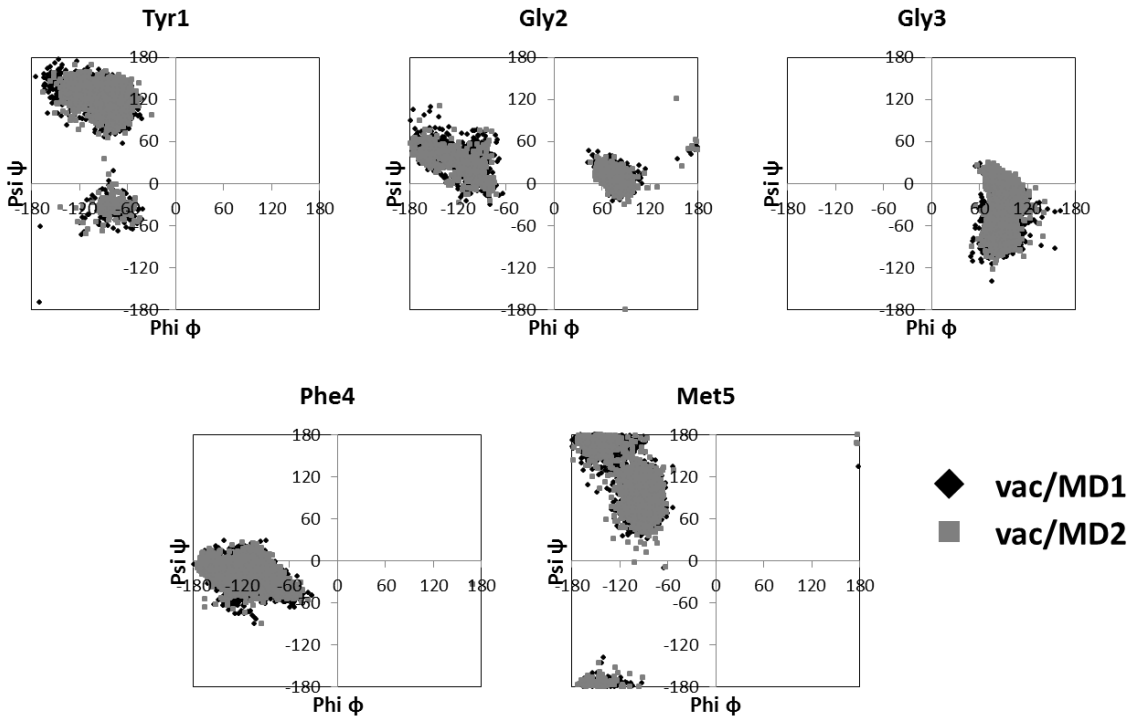


Figure 17: Ramachandran Plots for MET simulated in a vacuum.

The Ramachandran plots for both ensembles simulated in a vacuum (Figure 17) show one or two distinct clusters of points for both datasets. The Ramachandran plots for all simulations simulated in explicit solvent (Figure 18) show many more clusters of angles than the ensembles in the vacuum. [59,70] The two glycine residues (Gly2 and Gly3) behave similarly to free dipeptide (five clusters of ϕ and ψ),[74] which infers that even as a part of the peptide, the two Gly have free rotation as they would in isolation.[70,74] This result is congruent with what Malevanets and Wodak found in their multiple replica simulation repulsion for the two Gly in MET.[74] A visual comparison between the Ramachandran plots produced from the solvated MD/REMD simulations done here and the Ramachandran plots produced from the Multiple Replica Repulsion simulation, suggests occupation of similar regions on the Ramachandran plot.[74] Unlike Malevanets and Wodak's classical MD simulation, which covered less conformational space than the Multiple Replica Repulsion simulation, all simulations in this work (MD/REMD)

covered similar conformational space as the Multiple Replica Repulsion simulation. Malevanets and Wodak's classical MD simulation was run for 32 ns whereas the classical MD simulations in this work were run for 64 ns; the difference in conformational space might be due to the different simulation lengths.

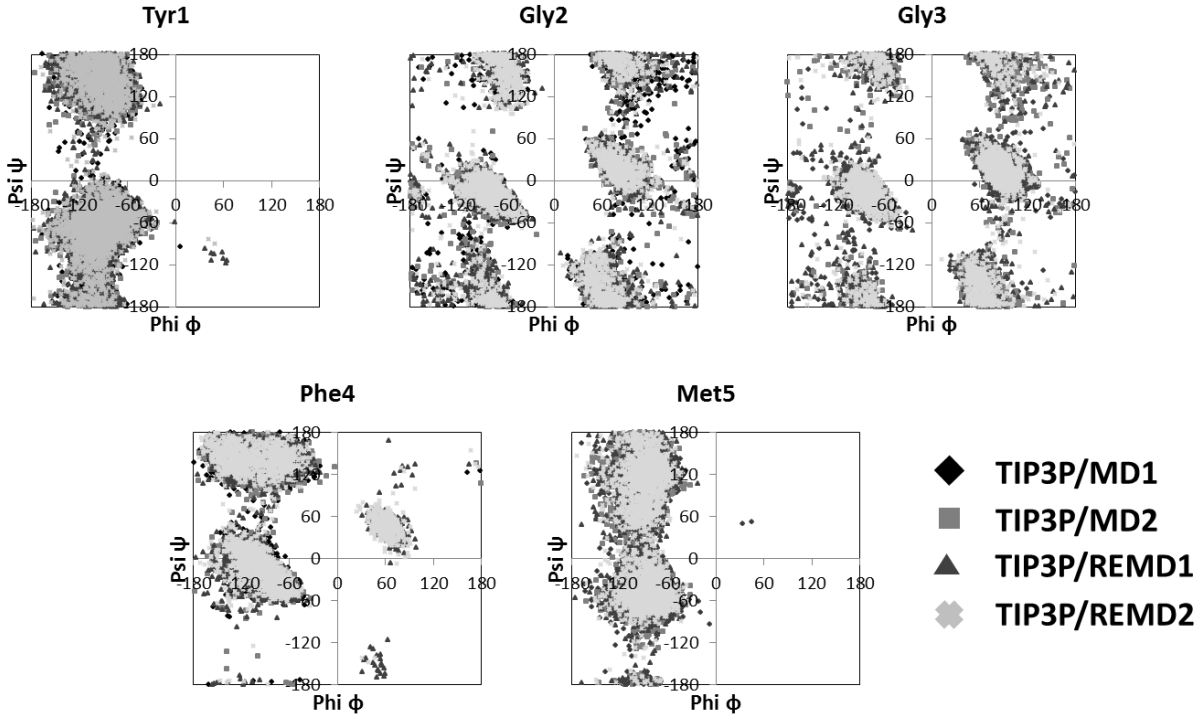


Figure 18: Ramachandran plots for MET simulated in a solution of TIP3P water

Two residues, Tyr1 and Phe4, explore more conformational space in the REMD simulations done here than the classical MD simulations (Figure 19). The REMD Ramachandran plot for Phe4 has a cluster at (φ,ψ) ~(70°, 40°) that is only occasionally explored during the classical MD simulations. A second cluster at (70°, -130°) sampled by the REMD simulations is not found in the solvated classical MD simulations. The REMD Ramachandran plot for Tyr1 has a cluster at (φ,ψ) ~(30°, -100°) that is not explored during the classical solvated MD simulations. Since more (φ,ψ) clusters are found for the Ramachandran plots of conformations sampled during the REMD simulations, these simulations covered more conformational space than the

classical solvated MD simulations. Although this method determined which simulation covered more conformational space (MD/REMD), this visual method could not quantify the degree of overlap between the different conformational distributions. Furthermore, this visual method could not determine which starting point covered more conformational space since the Ramachandran plots were too similar (i.e. did TIP3P/MD1 cover more or less conformational space than TIP3PMD2).

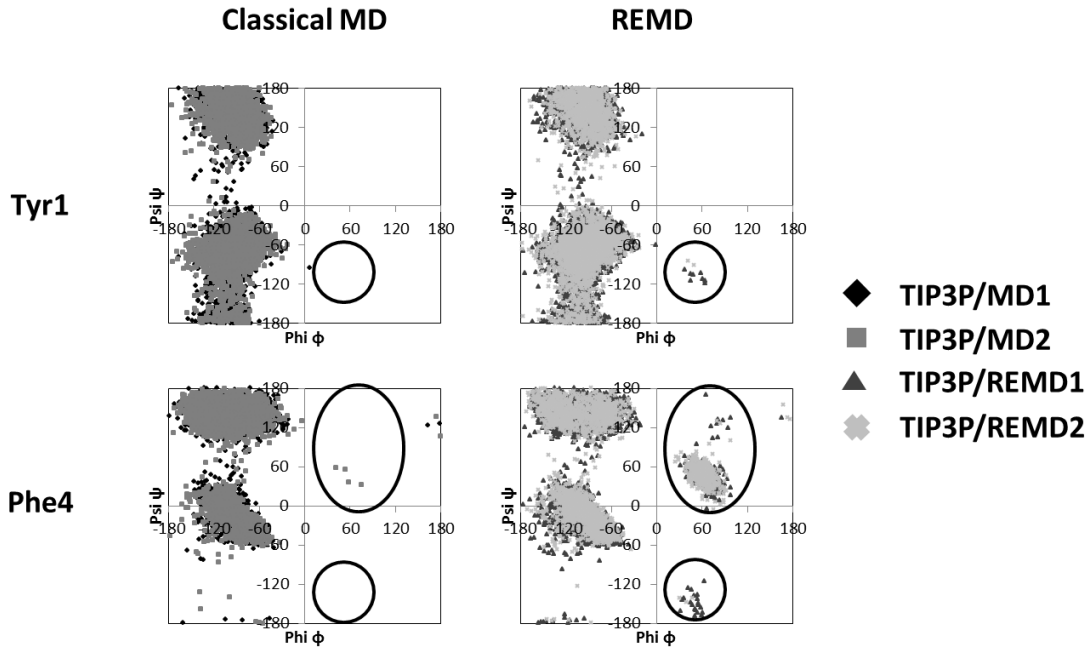


Figure 19: Ramachandran plots for MET simulated in TIP3P water by classical MD and REMD simulations. The plots to the left are for the classical MD (TIP3P/MD1 and TIP3P/MD2) and those on to the right are for the REMD simulations (TIP3P/REMD1 and TIP3P/REMD2). The circles show regions of (ϕ,ψ) where REMD covers more conformational space than classical MD. Specifics for the datasets are in Table 6.

7.3.2.2 Comparing Datasets of MET by SOMs

SOMs can be used to compare different datasets. Each node/neuron of a trained map has a nodal centre which describes, in an average way, the properties of the nodal membership. Each nodal centre can be visually compared to the nodal centres of other SOMs produced from clustering different datasets, or their numerical values can be compared.

7.3.2.2.1 Vacuum vs Solvated MET Ensembles

The SOM analysis began with the original C++ program, where three 5×5 SOMs were trained independently using both bordered and toroidal boundary conditions for the 20-dimensional datasets vac/MD1 and TIP3P/MD1 (Table 7: 7-u and 7-v). This produced a total of 12 SOMs (6 bordered and 6 toroidal). Figure 20 A shows an example of the toroidal map generated from the vac/MD1 dataset and Figure 20 B shows an example of the toroidal map generated from the TIP3P/MD1 dataset. The vertical axis is the percent occupancy, which is the percentage of conformations in one node from the total number of conformations in the dataset. The images surrounding the occupancy graphs are ribbon diagrams of the conformations of MET closest to the nodal center. By visual inspection of the ribbon diagrams for vac/MD1 (Figure 20 A), there appears to be little conformational diversity. The little conformational diversity is due to the lack of solvent molecules to screen the partial charges of MET in the vac/MD1 simulation. This result is congruent with the Ramachandran plots produced from the same dataset (Figure 17), showing fewer clusters of (φ,ψ) values since the peptide explores a smaller conformational distribution in a vacuum than the explicit solvent (TIP3P/MD1). When the same visual inspection was done for TIP3P/MD1 (Figure 20 B), the ribbon diagrams show reducing favourable intrapeptide electrostatic interactions and a much larger diversity. Solvent molecules screen the partial charges of MET, allowing more variation in the backbone conformations to take place. This result is congruent with Figure 18, where the Ramachandran plots show more values of (φ,ψ) angles sampled during the TIP3P/MD1 simulation.

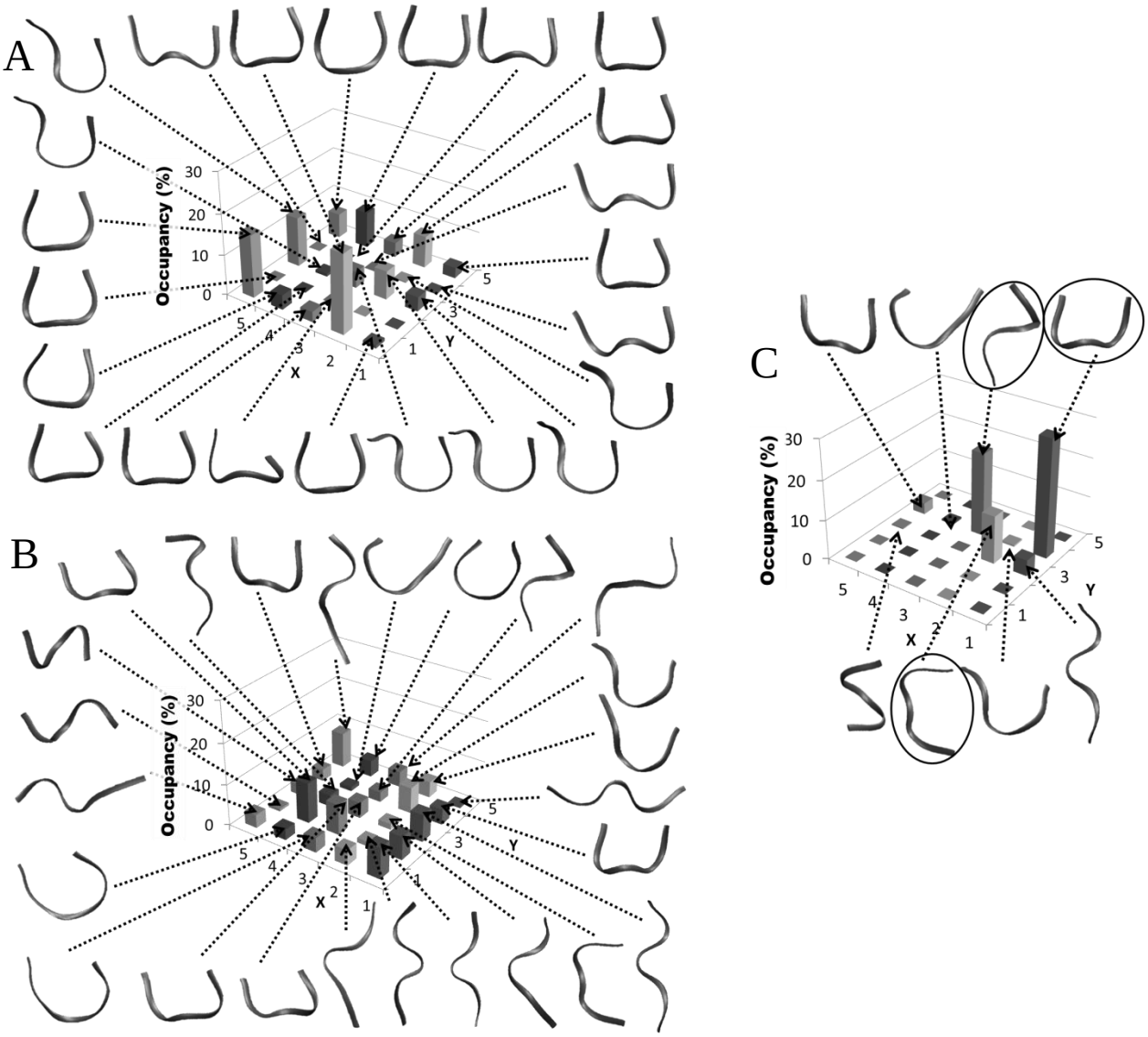


Figure 20: Occupancy of nodal clusters for a dataset of MET on a 5×5 SOM generated from the C++ program. Toroidal boundaries and consent neighbourhood were used during training. (A) vac/MD1 (Table 7:7-u) and (B) TIP3P/MD1 (Table 7:7-v). The peptide images are the conformations closest to the nodal centres. (C) The percentage of vac/MD1 dataset plotted on the TIP3P/MD1 map (B). The cutoff range is the data point (conformation) furthest from the nodal centre on the TIP3P/MD1 map.

From the Ramachandran plots (Figure 17 and Figure 18), as well as the visual comparison of Figure 20 A and Figure 20B, it is established that the dataset TIP3P/MD1 explored more conformational space than vac/MD1. However, this is done by visual comparison of two maps; with mapsizes greater than 5×5, this method of comparison is too time consuming and offers no real advantage over direct visualization of molecular trajectories. The SOMs

produced 20-dimensional vectors for each of the nodal centres, summarizing all the conformations in the nodal membership. A second vector (nodal "tolerance") is determined for each node which encompasses the dimensions of the conformation which is the most unlike the nodal center but still in the nodal membership. In combination with the nodal centres, these values of nodal tolerance make it possible to plot another dataset onto an already trained SOM, resulting in a supervised classification method (Figure 4).

The TIP3P/MD1 dataset was plotted onto the SOMs trained with the same dataset. Every single SOM had 100% of the conformations plotted onto the SOM generated from its own dataset (row 1 and row 4 in Table 8). This is the same for all supervised training of SOMs with their own datasets. The TIP3P/MD1 dataset was plotted onto the SOM trained using the vac/MD1 dataset (Table 8). Less than 2% of the conformations were allowed to join a nodal membership for any of the six maps produced (three bordered (1.2 ± 0.5%) and three toroidal (0.9 ± 0.4%) for the vac/MD1 dataset,). However, when the vac/MD1 dataset was plotted on the SOMs trained using the TIP3P/MD1 dataset, ~90% (Figure 20 C) of the conformations were able to be classified as being similar to a cluster of TIP3P/MD conformations (Table 8), for any of the six maps produced (three bordered (88.1 ± 1.4%) and three toroidal (85.5 ± 5.0%) for the TIP3P/MD1 dataset). Figure 20 C shows ribbon diagrams corresponding to the eight clusters of vac/MD1 conformations. The majority of the conformations (82.9%) are gathering in three nodes (49.3%, 21.4%, and 12.2%). The percentage of data accepted onto a map obtained by supervised clustering is a quantitative way to compare ensembles.

Table 8: The percentage of the dataset plotted on a 5×5 SOM generated from another dataset. The data shown is using the same parameters other than boundaries.

Datasets	Maps Generated from Datasets	Percentage of Data on Map Rectangular Bordered		Rectangular Toroidal	
vac/MD1	vac/MD1	100.0	±0.0	100.0	±0.0
	TIP3P/MD1	88.1	±1.4	85.5	±5.0
TIP3P/MD1	vac/MD1	1.2	±0.5	0.9	±0.4
	TIP3P/MD1	100.0	±0.0	100.0	±0.0

Averages and standard deviations are computed from 3 different SOM with boundaries above the parameters are 7-u and 7-v (Table 7).

7.3.2.2.2 Solvated vs Solvated MET Ensembles

To evaluate the overlap of the conformational samples collected from the simulations of the solvated MET, each of the datasets were classified by SOMs trained using all the solvated datasets: TIP3P/MD1, TIP3P/MD2, TIP3P/REMD1, and TIP3P/REMD2. When a dataset is clustered using its own SOM as a supervised classifier, 100% of the dataset should be assignable. If two independent computer simulations have explored the same conformational space, then a finite conformational sample from one simulation ought to be described by an SOM trained using a conformational sample from the second. Table 9 shows the average percentage of conformations assigned to nodal memberships of SOMs trained by different datasets, as well as their corresponding standard deviations. The averages are calculated from 100 SOMs. These SOMs were trained using identical parameters other than their boundaries and/or nodal geometry. For example, 91.3 ± 4.3% of the 3200 conformations saved from the TIP3P/MD1 simulation are able to be clustered by 100 rectangular bordered SOMs trained independently using the TIP3P/MD2 3200 member conformational dataset. The percent of TIP3P/MD1 data assigned to SOMs trained using TIP3P/MD2 data was the same within one standard deviation for rectangular or toroidal boundaries and for rectangular and hexagonal nodes (91.3 ± 4.3%, 89.4

± 4.4%, 91.6 ± 5.0%, and 88.7 ± 4.3%). Examination of Table 9 suggests that all simulations of solvated MET sampled similar regions of conformational space, with ~83%-95% overlap. Figure 21 is a pictorial representation of the overlap of conformational distributions the simulations covered. This result is congruent with the Ramachandran Plots (Figure 18), where the datasets cover very similar (φ, ψ) space. Furthermore, there is a higher percentage when the datasets are plotted on SOMs generated from the REMD simulations, though this is only statistically significant for some of the SOMs. The percentage of TIP3P/REMD2 plotted on hexagonal bordered SOM trained by TIP3P/MD1 is 84.7 ± 4.3% which is less than that for the same data plotted on an identical SOM trained from TIP3P/REMD1 (94.0 ± 3.0%).

Table 9: The percentage of conformational datasets plotted on previously trained 10×10 SOMs. The results showed used the same parameters other than geometry and boundaries.

Datasets	Maps Generated from Datasets	Percentage of Data Assigned to Maps							
		Rectangular Bordered		Rectangular Toroidal		Hexagonal Bordered		Hexagonal Toroidal	
TIP3P/MD1	TIP3P/MD1	100.0	±0.0	100.0	±0.0	100.0	±0.0	100.0	±0.0
	TIP3P/MD2	91.3	±4.3	89.4	±4.4	91.6	±5.0	88.7	±4.3
	TIP3P/REMD1	95.2	±3.3	94.3	±6.7	95.0	±4.2	94.2	±4.9
	TIP3P/REMD2	94.5	±4.0	91.6	±5.6	92.0	±3.5	94.0	±4.1
TIP3P/MD2	TIP3P/MD1	92.9	±4.0	91.2	±4.6	91.7	±4.5	89.4	±4.6
	TIP3P/MD2	100.0	±0.0	100.0	±0.0	100.0	±0.0	100.0	±0.0
	TIP3P/REMD1	94.0	±3.9	93.6	±6.2	93.4	±4.8	92.8	±5.7
	TIP3P/REMD2	94.4	±4.3	91.8	±4.9	93.1	±3.8	93.8	±4.3
TIP3P/REMD1	TIP3P/MD1	90.9	±3.4	86.1	±5.1	85.1	±4.9	84.1	±4.6
	TIP3P/MD2	91.5	±3.0	86.6	±4.9	90.1	±2.6	84.5	±6.7
	TIP3P/REMD1	100.0	±0.0	100.0	±0.0	100.0	±0.0	100.0	±0.0
	TIP3P/REMD2	95.8	±3.0	91.5	±4.6	92.3	±2.9	93.2	±3.6
TIP3P/REMD2	TIP3P/MD1	90.8	±3.2	85.4	±5.0	84.1	±4.3	83.1	±4.6
	TIP3P/MD2	90.9	±3.1	85.7	±5.0	90.0	±3.1	83.9	±6.3
	TIP3P/REMD1	95.3	±2.0	92.5	±5.0	94.0	±3.0	92.9	±4.2
	TIP3P/REMD2	100.0	±0.0	100.0	±0.0	100.0	±0.0	100.0	±0.0

Averages and standard deviations are computed from 100 different SOM with geometry and boundaries above the parameters are 7-w to 7-dd (Table 7).

The ability for SOMs to compare independently generated ensembles makes them a useful tool in analysing MD and REMD simulations. The protocol for using SOMs as a supervised clustering is novel for comparing simulated ensembles. Unlike Ramachandran plots, this method considers the entirety of the conformation (i.e. all dihedrals at once) and can be done with more complex (i.e. larger) proteins.

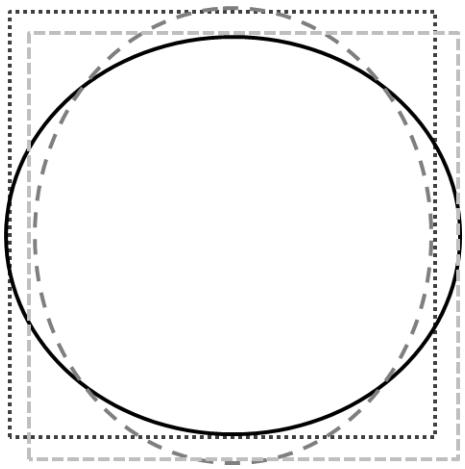


Figure 21: A pictorial representation of the overlap of the different simulations from Table 9. (—) is TIP3P/MD1, (–) is TIP3P/MD2, (…) is TIP3P/REMD1 and (– –) is TIP3P/REMD2

7.3.3 Edge Effect

The bordered map has what is known as an "edge effect", which is caused by conformations collecting on the edge nodes of the bordered map.[80] Bordered maps have fewer nodes in the neighbourhood when the BMU is on the edge of the map (Table 1). This means the nodes on the edge of the map have fewer surrounding nodes contributing to the updating of their nodal centres, than do nodes in the interior. This causes conformations to get "stuck" on the edge of the map, resulting in larger clusters that may not be well separated in their characteristics from other nodes. Toroidal maps do not have this problem, since all nodes have the same number of neighbours no matter the location on the map. Bordered maps have significantly more conformations on the edge than the interior nodes, while toroidal maps show no difference.

To demonstrate the edge effect and to determine how the size of the map affects the edge effect, the average number of conformations collected in nodes on the edge of the trained SOM versus those in interior nodes of the SOM was plotted (Figure 22). This calculation was done for both bordered and toroidal SOMs, where the designation of what is considered edge and interior node was derived from the bordered map. These two values for the set of SOMs were normalized by dividing by the total number of conformations and multiplying by the number of nodes in the SOM. If the toroidal boundaries were implemented correctly, the normalized value should be ~1, since all locations on the map are equally likely for all clusters. If the bordered maps have an edge effect, the populations in the edge nodes should be statistically higher than that of the interior nodes. Figure 22 shows data from SOMs described as 7-a, 7-ee, and 7-ff (Table 7) trained using the TIP3P/MD1 dataset. There are 500 rectangular SOMs in each case. The C++ program was used and the training parameters were the same. As predicted, the toroidal SOMs have a normalized frequency of ~1 meaning that the toroidal boundaries were implemented correctly. Furthermore, the edge nodes in the bordered map contain more conformations than the interior nodes, showing that there is an edge effect in bordered SOMs. The normalized frequency also shows that as the map increases in size, the edge effect also increases, in that the proportion of conformations placed in the edge nodes increases from 5×5 to 7×7 to 10×10 maps.

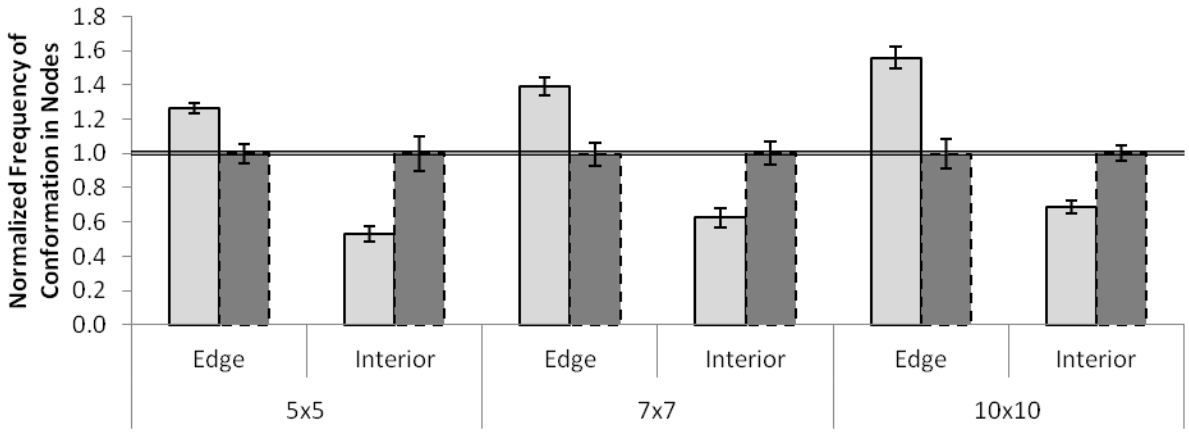


Figure 22: The normalized frequency of conformations contained in either edge or interior nodes. The designation of these nodes is from the bordered SOM. ☐ is the bordered SOM and ▩ is the toroidal SOM. This graph used trained SOMs described in Table 7:7-a, 7-ee, and 7-ff.

7.3.3.1 Using the Objective Function (INSSQ) to Show the Edge Effect

The within group sum of squares (INSSQ), from equation [1], for bordered, toroidal and the untrained maps were compared to each other (Figure 23). As expected, the untrained map has larger values of INSSQ than either the bordered or toroidal maps on the histogram. As the mapsizes increases, the ordering of the histograms generated from the objective function (INSSQ) changes from toroidal-bordered to bordered-toroidal. For the 5×5 and 7×7 maps, the toroidal map clusters the conformations better than the bordered map (lower values of INSSQ: Figure 23A and Figure 23B). However, the 10×10 bordered map clusters the conformations better than the toroidal since it produces tighter clusters (Figure 23C).

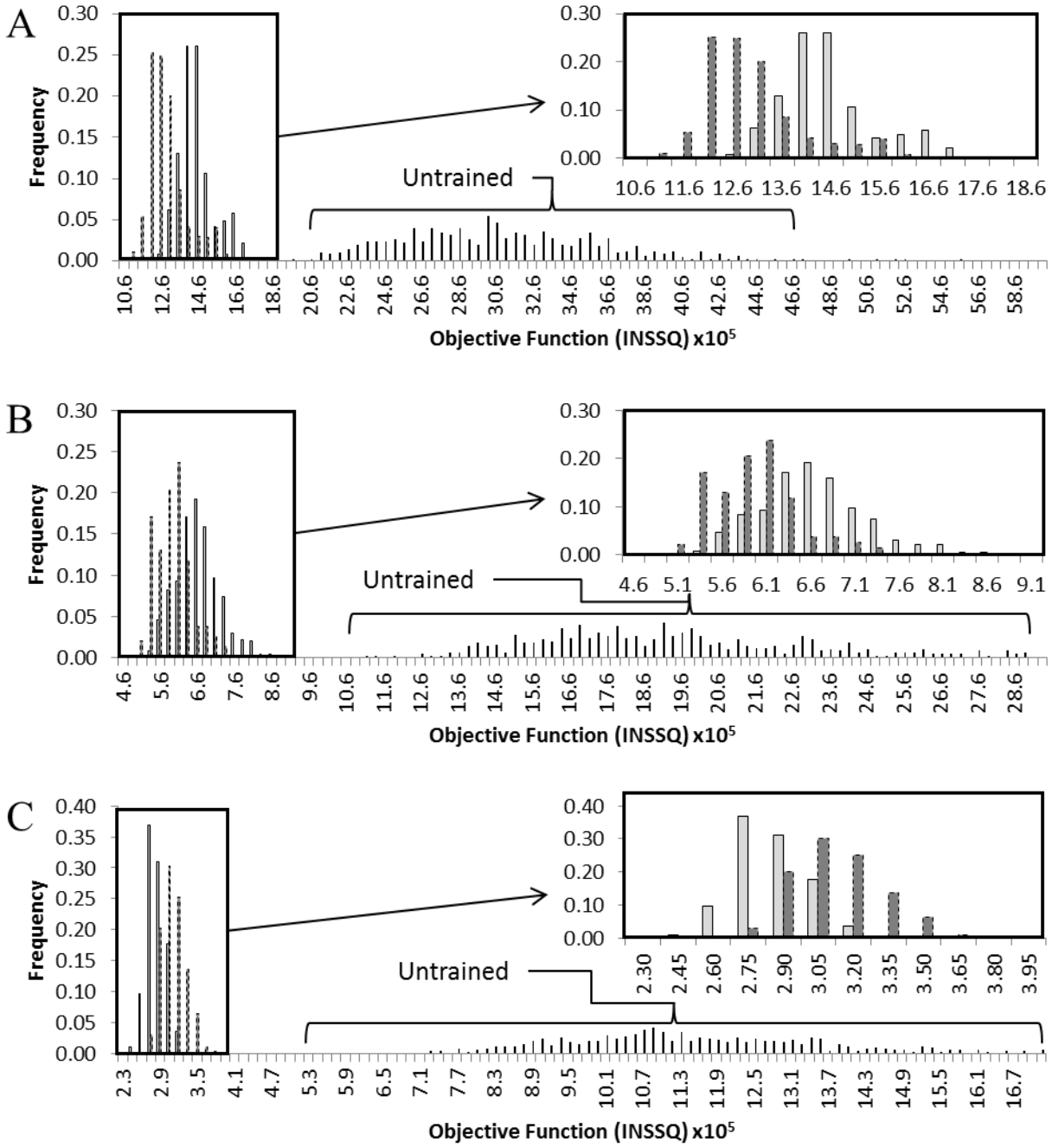


Figure 23: Histograms for the normalized frequency of the Objective Function (within group sum of squares; INSSQ), for the ■ untrained, □ bordered, and ▩ toroidal rectangular SOMs. (A) 5×5 (Table 7:7-a and 7-d), (B) 7×7 (Table 7:7-ee and 7-g), and (C) 10×10 SOMs (Table 7:7-ff and 7-i). The values for bordered and toroidal maps have been enlarged in the upper right insert. There are 500 SOMs for each map type.

To discover why the relative order of the magnitude of INSSQ changes between bordered and toroidal when the mapsize changes from 7×7 to 10×10 maps, the trained SOM is broken into three sections (corner, inner edge and interior bins) and the INSSQ was calculated for these

sections (Figure 24). These sections were all designated by the bordered SOMs. When INSSQ was calculated for the three sections of the toroidal maps, they overlap, showing no preference with respect to geometry. However, when the same calculations were performed for the bordered maps the contents of interior bins produce a smaller objective function than the edge bins, and edge bins produce a smaller objective function than the corner bins. The bordered 10x10 map has so many of the conformations in the interior bins that it is these groups which contribute to the majority of the overall INSSQ (Figure 25).

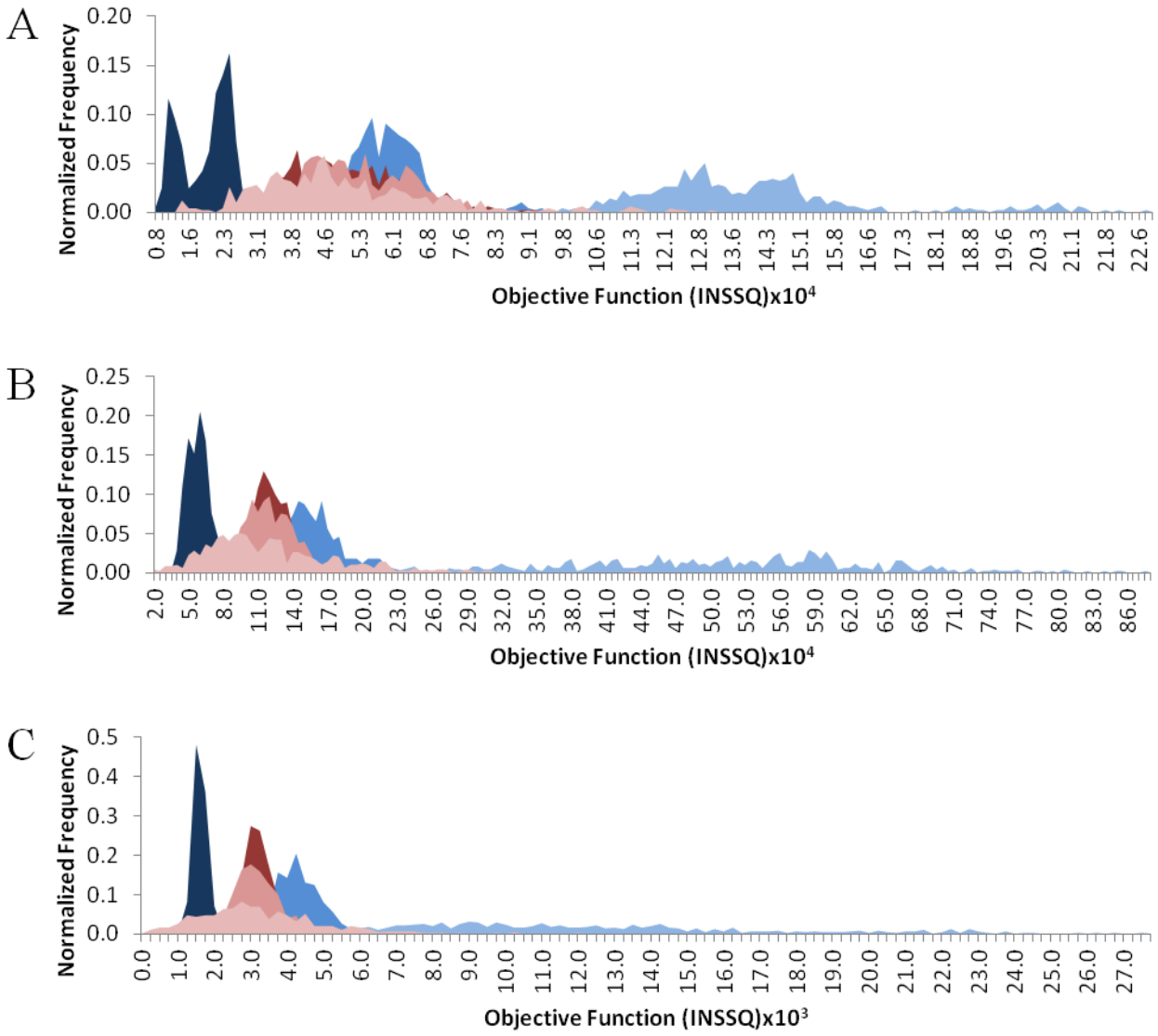


Figure 24: The INSSQ objective function histograms for the normalized frequency of the three sections (interior, edge, and corner) of the bordered (blue) and toroidal (red) rectangular. Bordered interior nodes, Bordered inner edge nodes, Bordered corner nodes, Toroidal interior nodes, Toroidal inner edge nodes, and Toroidal corner nodes. (A) 5×5 (Table 7:7-a), (B) 7×7 (Table 7:7-ee), and (C) 10×10 (Table 7:7-ff). There are 500 SOMs for each map type.

From Figure 22 it was determined that the edge effect increases as the mapsize increases. From Figure 23 it was determined that the relative order in the value of the objective function INSSQ changes when the mapsize changes from 7×7 to 10×10 bordered maps. Figure 24 shows that this could be due to the proportion of interior nodes to exterior nodes. To determine if this is the case, Figure 25 only shows node contributions of the interior to the objective function for

both boundary conditions (bordered and toroidal). When the size of the map increases, there are more nodes in the interior of the SOM than the exterior of the SOM, resulting in a higher probability that conformations will be placed in an interior node as the mapsize increases. Bordered maps can cluster data; however, toroidal maps cluster data better since they do not have an edge effect. This method with INSSQ is one way to prove toroidal maps cluster data better than bordered maps.

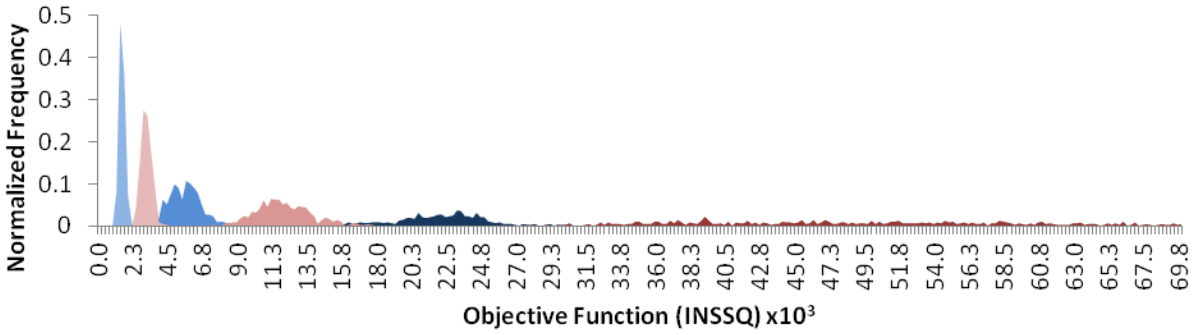


Figure 25: The INSSQ objective function distribution of interior nodes of different sized SOMs and different geometries. The histograms for the normalized frequency of the bordered and toroidal interior nodes of the rectangular SOMs for the Objective Function (INSSQ). bordered 5×5, bordered 7×7, bordered 10×10, toroidal 5×5, toroidal 7×7, and toroidal 10×10. There are 500 SOMs for each map type. This graph is from the data generated from SOMs in Table 7:7-a, 7-ee, and 7-ff.

7.3.3.2 Using C_v to Show the Edge Effect

The dataset TIP3P/MD1 was used to generate 500 independently trained SOMs that were compared to each other by C_v, resulting in 500^2 C_v values. The highest C_v value of 1.0 is obtained when the data is partitioned in exactly the same groupings (i.e. an SOM compared to itself). The C_v was computed between maps that have the same mapsize and parameters. A normalized histogram of C_v values was generated for each of following comparisons: untrained to untrained, bordered to bordered, toroidal to toroidal, and bordered to toroidal SOMs. Figure 15 (Section 7.3.1) illustrates the C_v distributions for untrained SOMs using the maxmin initialization and different mapsizes and geometries. The 5×5 rectangular untrained SOM generated from the

C++ program (Table 7: 7-d) has a peak in the C_v distribution at ~0.30 (Figure 26) which falls at much lower values than the C_v distributions for trained maps. The partitioning of the MET conformations on the untrained maps is very different from each random initialization. The C_v distributions for comparisons of bordered to bordered trained maps has a peak at ~0.70 and has a right tail skewed to another peak at ~0.91. The lower peak, which we have labeled the majority complement region, is when different SOM partitionings are compared. However, the different partitionings in trained SOMs are still more similar than untrained SOMs because conformations which are similar are placed together whereas they will be placed in different nodes on untrained SOMs. The higher peak, which we have labeled the high C_v region, is when different SOMs having very similar partitionings are compared, these are deemed reproducible SOMs. The C_v distribution of toroidal to toroidal SOMs has two peaks at ~0.75 and ~0.92, with almost no intervening values. The C_v distribution for bordered compared to the toroidal SOMs has a peak at ~0.68 and no C_v values above 0.75. The lower values of C_v for bordered to toroidal SOM comparisons shows the SOMs with these two different boundary conditions are not very similar. All the bordered maps have an edge effect (Figure 22); this is influencing the contents of the clusters. Bordered maps are more alike to each other than to toroidal SOMs and toroidal maps are more alike to each other than to bordered maps. When bordered maps are compared to themselves there is a peak in the high C_v region at ~0.91; similarly, when toroidal maps are compared to each other there is a peak in the high C_v region at ~0.92. However, when bordered maps are compared to the toroidal maps there is no high C_v region. This means that the partitioning of bordered maps is not reproduced on the toroidal maps and vice versa. The toroidal SOM produces better clusters that are somewhat more reproducible, since there are more C_v values with values >0.9.

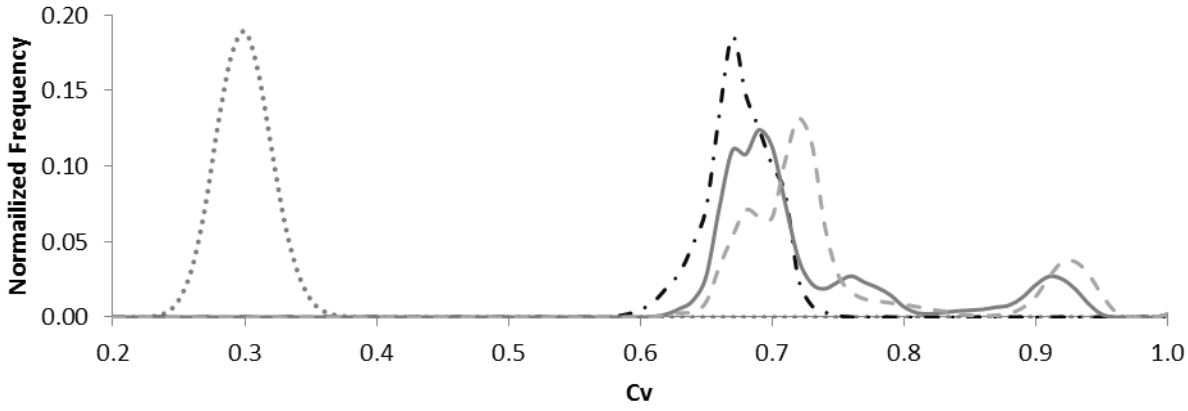


Figure 26: The C_v distributions for the (trained and untrained) C++ 5×5 rectangular SOMs. The sequentially trained by the Table 7:7-a and the untrained SOMs by Table 7:7-d. The four different distributions are of (....) untrained to untrained, (——) bordered to bordered, (– –) toroidal to toroidal, and (—.) bordered to toroidal SOMs.

7.3.4 Reproducible Clusters in the C++ Program

Clustering validity is also assessed by the reproducibility of data partitioning using the same and different algorithms.[20,33] Most partitional clustering algorithms must be run multiple times to find the optimum partitioning of the data.[20] The goal of data partitioning usually is to find the optimum number of clusters (i.e. fewest number of clusters); this is usually a cluster value of fewer than 10.[33] The SOMs in this study all have more than 10 clusters; this is advantageous because flexible peptides and proteins have broad conformational distributions that would be better described by a larger number of conformational clusters. SOMs have not been often used to cluster molecular conformations obtained from computer simulations.[4,16,37–39] This may be due, in part, to the difficulty in comparing the different partitionings, produced by repeated cluster initializations, of a dataset into nodal memberships of an SOM. The SOMs in this study have been run multiple times to evaluate the extent to which reproducible clusters are produced. This section will focus on the C_v distributions for the C++ toroidal 5×5 rectangular SOMs (Table 7:7-a) in Figure 26 and will evaluate these SOM partitionings by means of an objective function.

The NEIGH objective function (equation [3]) is analogous to the INSSQ objective function (equation [1]) because both objective functions give a numerical value which assesses the quality of data clustering. A better cluster would have a lower objective function value. The NEIGH objective function evaluates the partitioning by examining the differences in dihedral angle of the conformations in the nodes as well as the dihedral angles of the conformations in the neighbourhood of that particular node; this evaluates the partitioning in a similar manner to how the map was trained. The SOMs were run multiple times (500), ensuring a higher probability of generating maps with similar partitioning of the data. Figure 27 further evaluates the toroidal 5×5 (25 clusters) rectangular SOMs analyzed in Figure 26 (the same C_v distribution is seen at the bottom of Figure 27).

Figure 27 uses a scatter plot of the NEIGH objective functions versus the C_v values calculated with that particular map. This means that one C_v value has two NEIGH objective functions because a C_v value is calculated between two SOMs. In Figure 27 the scatter plot contains 250000 data points because each SOM is compared to 500 SOMs (including itself). The goal of this analysis is to find reproducible results (i.e. a low objective function and a high C_v value; this produces similar maps which have tight clusters). However, not all maps which produce a high C_v value will be similar to all other maps which produced high C_v values. There can be different partitionings of the data which could be reproducible. A program was written to compare SOM pairs with high C_v values to determine which SOMs contained nodes with similar content. When the C_v values are computed between, say, three maps that have very similar partitionings, all of these SOM comparisons (i.e. three pairs) should produce high C_v values. The program started with the top 25% of high C_v values and was expanded to include all comparisons of a particular partitioning; these different partitionings formed groups. Figure 27 also highlights

the specific groups found by the program that have three or more SOMs in a group. There are 21 groups shown on this graph. The largest group contains 184 SOMs (37% of the 500 total) and has overall low objective function values and is therefore deemed reproducible. Due to the high percentage of SOMs in this group, it was determined that generating 500 maps was excessive and producing 100 SOMs would still create this data partitioning many times (i.e. ~37 of 100 maps).

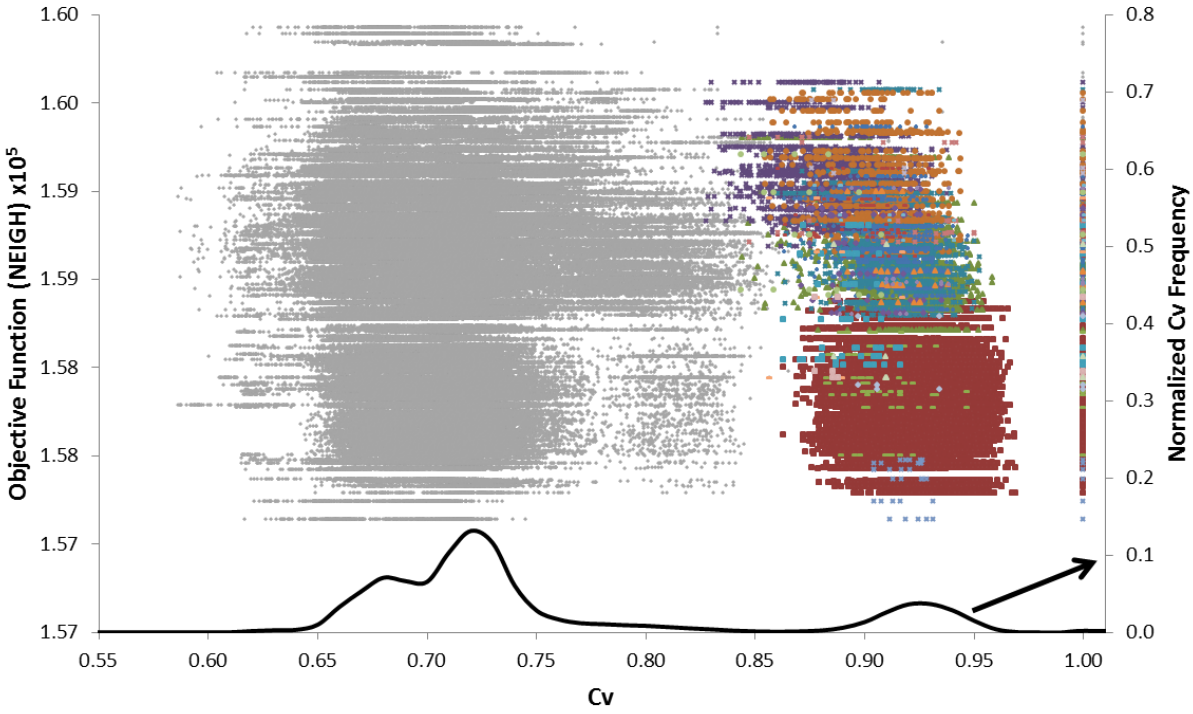


Figure 27: The NEIGH objective function versus C_v values for the 500 toroidal 5×5 rectangular SOMs trained by the TIP3P/MD1 dataset from the C++ program (Table 7:7-a). The scatter graph is the NEIGH objective function (left axis) against C_v (x-axis). When a C_v distribution is generated from the scatter graph (—), the distribution is a duplicate of the same dataset in Figure 29 (normalized C_v frequency; right axis). The C_v is generated from two maps; for each map a NEIGH objective function was computed (one C_v value has two NEIGH objective functions). ◆ are all the C_v values, the coloured points are the 21 groups of maps which produce high C_v values with each other and have 3 or more maps in a group. ■ is the highest occupied group with 37% of the SOMs in this group.

Figure 28 is similar to Figure 27 where the C++ toroidal 5×5 rectangular SOMs (Table 7:7-a) are examined by an independent comparison method. While Figure 27 uses the C_v to compare SOMs, Figure 28 A uses SI (equation [24]) and Figure 28 B uses rSI (equation [25])..

Figure 28 A has no discernible pattern linking the top ~25% of SI values to the NEIGH objective function. Therefore, SI is not a good method for comparing SOMs. However, when the rSI is used to group partitionings of the SOMs together, it independently produced similar groups as the C_v. Furthermore, the group with the highest occupancy in both Figure 27 and Figure 28B is very similar. The rSI highest occupancy group contains one fewer map than the C_v highest occupancy group (all other SOMs are the same). These independent evaluations of the partitioning produce the same grouping of reproducible maps. This validates this new comparison method. However, rSI is more computationally expensive than C_v and since similar results are obtained by both comparison methods, the C_v comparisons will be used throughout the rest of this work.

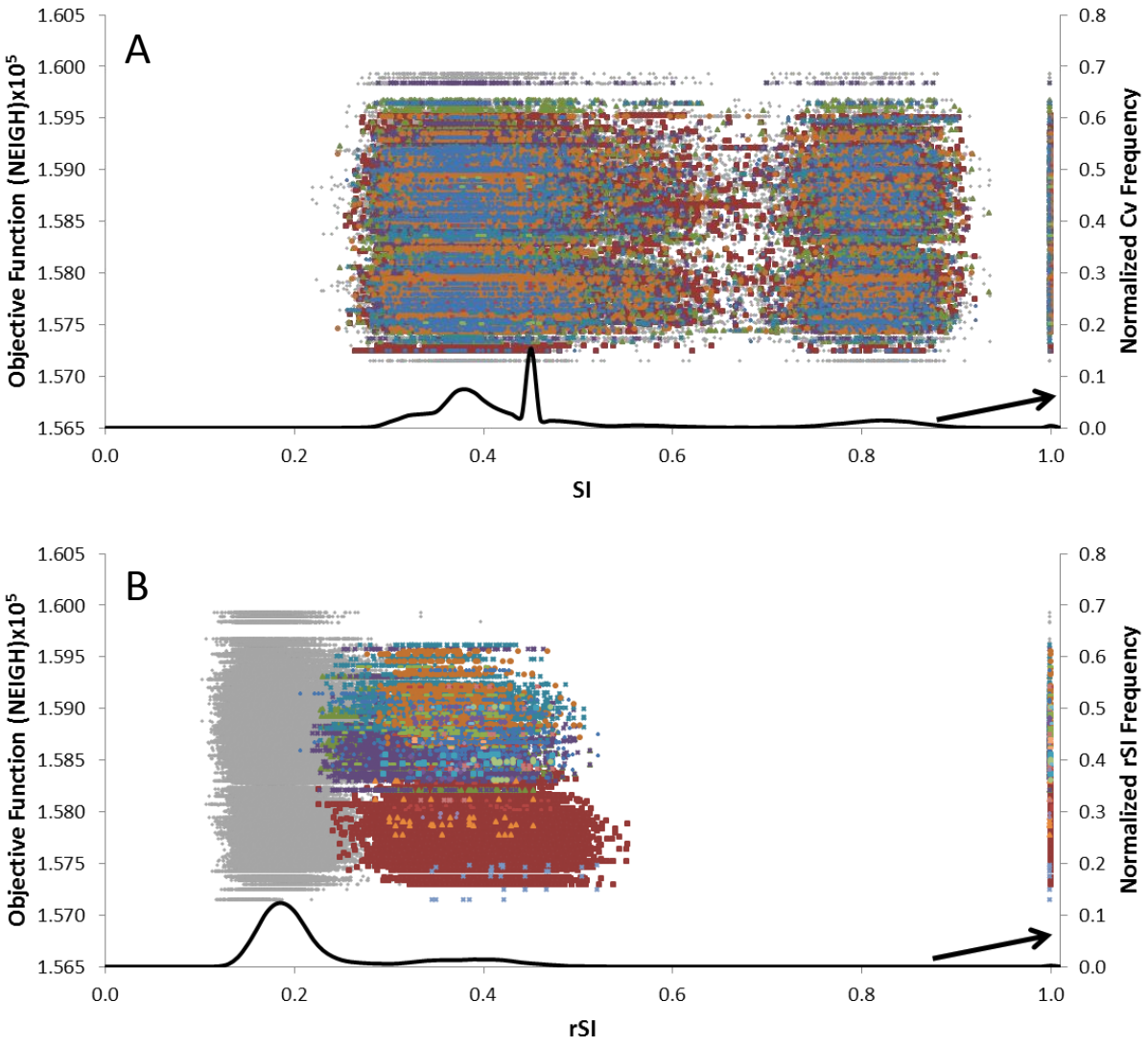


Figure 28: The NEIGH objective function versus SI and rSI values for the 500 toroidal 5×5 rectangular SOMs trained by the TIP3P/MD1 dataset from the C++ program (Table 7:7-a). (A) scatter graph of NEIGH objective function (left axis) against SI (x-axis). (B) scatter graph of NEIGH objective function (left axis) against rSI (x-axis). A distribution of the x-axis was generated from the scatter graph (—) and the normalized frequency is on the right axis. ◆ are all the values, the coloured points are the 8 groups in SI and 17 groups in rSI which produce high values with each other and have 3 or more maps in a group. For (B) ■ is the highest occupied group with 37% of the SOMs in this group.

7.3.5 Comparing Independent Source Code for Generating SOMs

The C++ SOM program was the starting point for analysing the conformations obtained from MD simulations. However, using one program to generate SOMs did not give us confidence that the same conclusions can be made for all SOM programs. To determine if SOM

reproducibility is program dependent/independent, another SOM program with an independent source code was used; the program is *somtoolbox* 2.0 found in MATLAB. The parameters used in the MATLAB program in this section are either the default settings or the preprogrammed settings which were used to mimic those of the C++ program.

7.3.5.1 Mapsize and Nodal Geometry by the C_v

The C_v distributions of the trained rectangular geometry SOMs need a different comparison method than the untrained maps (Figure 15), since it is unclear how the mapsize or nodal geometry affects the partitioning. In order to compare the effect of mapsize on the C_v distribution, multiple C_v distributions would need to be generated by SOMs with different mapsizes. Figure 29 shows the C_v distibutions generated from the 5×5 and the 7×7 SOMs, different geometries (bordered and toroidal) and from two different programs (C++ and MATLAB). The ideal solution is where all SOMs give a C_v value of ~1. However, this is unrealistic due to the different initializations of the SOMs. The next best solution is where there are two peaks; a high C_v region and a majority complement region (Figure 26). This means there are maps which have similar partitionings. The C++ program for the 5×5 SOMs (Table 7: 7-a) generated two peaks in the C_v distributions for both the bordered and toroidal SOMs (same C_v distributions seen in Figure 26). One peak is located around ~0.7, where the majority complement of C_v comparisons are located. The other peak is located in the high C_v region, for both bordered and toroidal SOMs. These peaks are highlighted with ⏃. However, when the mapsize increases to 7×7 (Table 7: 7-ee) there is only a singular peak located around ~0.7 (shown with ⏃). The training algorithm for the C++ program employed a Gaussian function that decreases with each iteration for both the learning rate function (equation [13] to [17]) and the neighbourhood radius function (equation [8] to [10]) while the MATLAB program employs a

decreasing inverse function (equation [15]) for the learning rate function and decreasing linear function (equation [9]) for the neighbourhood radius function. The different neighbourhood function for the MATLAB program generated toroidal maps that have consistently higher C_v values for the larger 7×7 maps. One of the artifacts of the different functions being used is that the bordered SOMs compared to bordered SOMs and toroidal SOMs compared to toroidal SOMs for the C++ program are more similar than the SOMs generated from the MATLAB program. For the C++ program, both boundary conditions produced more similar maps (C_v ~0.65-0.80) than did the MATLAB program for bordered (C_v ~0.55-0.70) but not toroidal boundaries (C_v ~0.75-0.95). The MATLAB program produces SOMs with a larger variation in the C_v distributions when comparing the different boundary functions than does the C++ program. The majority complement of C_v comparisons values for the MATLAB bordered SOMs are lower than those for the toroidal SOMs. The C++ program produces C_v distributions for the different geometries which overlap in their majority complement regions. The variations in the SOM programs can be seen as the mapsize increases from 5×5 to 7×7. The C++ program does not produce SOMs that vary much with the different boundary conditions (bordered and toroidal) in their majority complement of C_v comparisons and the high C_v region disappears when the mapsize is larger than 5×5.

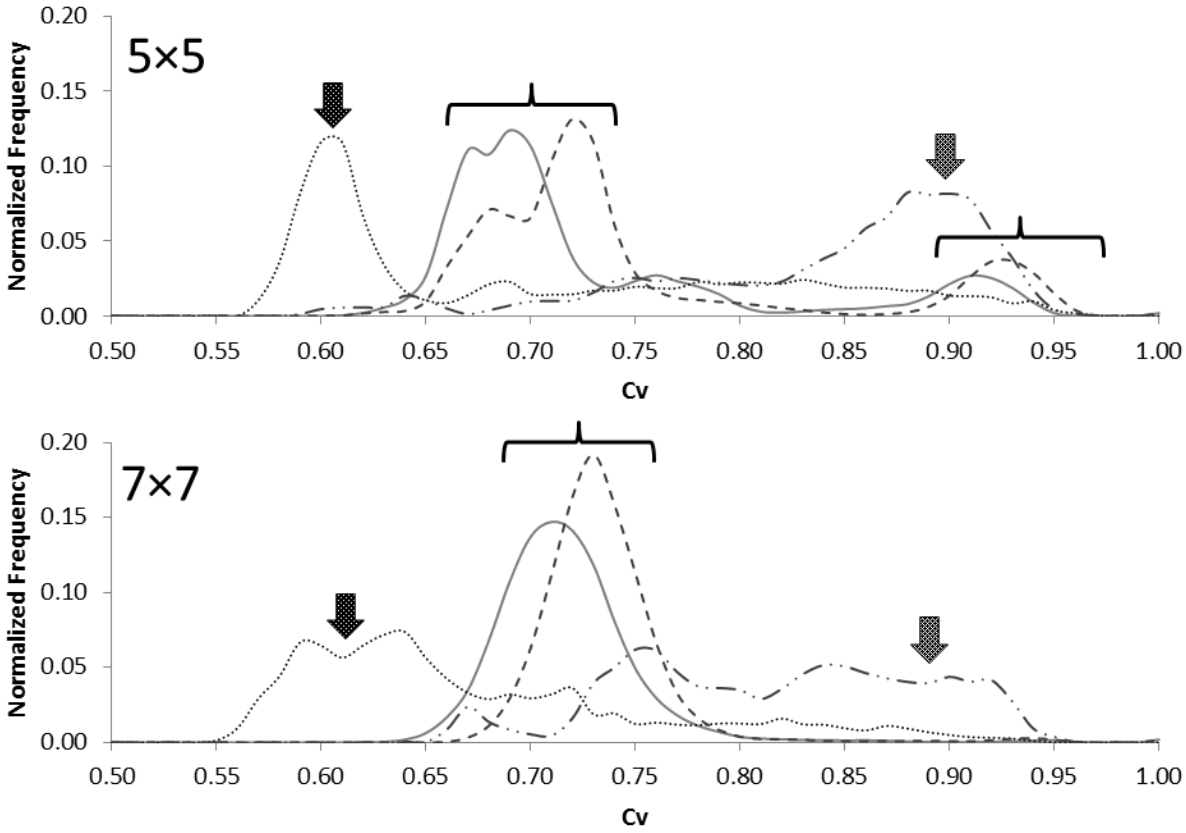


Figure 29: The C_v distributions for 5×5 and 7×7 sequentially trained SOMs of the same parameters. The maps are separated based on their mapsize. The lines are organized based on the boundaries and the training program (C++/MATLAB). ⬇ illustrates the majority complement of C_v comparisons of the bordered rectangular SOM from the MATLAB program and ⬇ is the majority complement of C_v comparisons of the toroidal rectangular SOM from the MATLAB program. ⌐¬ illustrates the peak of both bordered and toroidal rectangular SOMs from the C++ program. (—) bordered rectangular SOM from the C++ program (Table 7: 7-a and 7-ee), (……) bordered rectangular SOM from the MATLAB program (Table 7: 7-gg and 7-hh), (– –) toroidal rectangular SOM from the C++ program (Table 7: 7-a and 7-ee), (—) toroidal rectangular SOM from the MATLAB program (Table 7: 7-gg and 7-hh).

Figure 30A shows the C_v distribution for 10×10 rectangular SOMs generated from the C++ program (Table 7:7-ff) and the MATLAB program (Table 7:7-ii). Neither group of SOMs produces C_v values in the upper limit (C_v ~0.9) suggesting that none of the SOMs are extremely similar. The values remain close to the majority complement of C_v comparisons region. Figure 30B shows the C_v distributions of the 10×10 SOMs generated from the MATLAB program (Table 7:7-ii and 7-jj) where the maps have different nodal geometries (rectangular and hexagonal). The C_v distributions for the 10×10 rectangular SOM produced from the MATLAB

program is repeated in Figure 30A and Figure 30B; the arrows show the majority complement of C_v comparisons in both graphs. The other C_v distributions are for the hexagonal 10×10 SOMs with both boundary conditions (bordered and toroidal) generated from the MATLAB program (Table 7:7-jj). The hexagonal map has more nodes in its neighbourhood than the rectangular map. The increase of neighbouring nodes could account for the right tail skew in the bordered SOM and the broader distribution of majority complement of C_v comparisons in the toroidal SOM. The broad distribution of the majority complement of C_v comparisons in the toroidal SOM comes close to the upper limit of the C_v comparisons. This broad C_v distribution for the hexagonal maps probably means that there are more maps that represent similar partitioning, which is desirable.

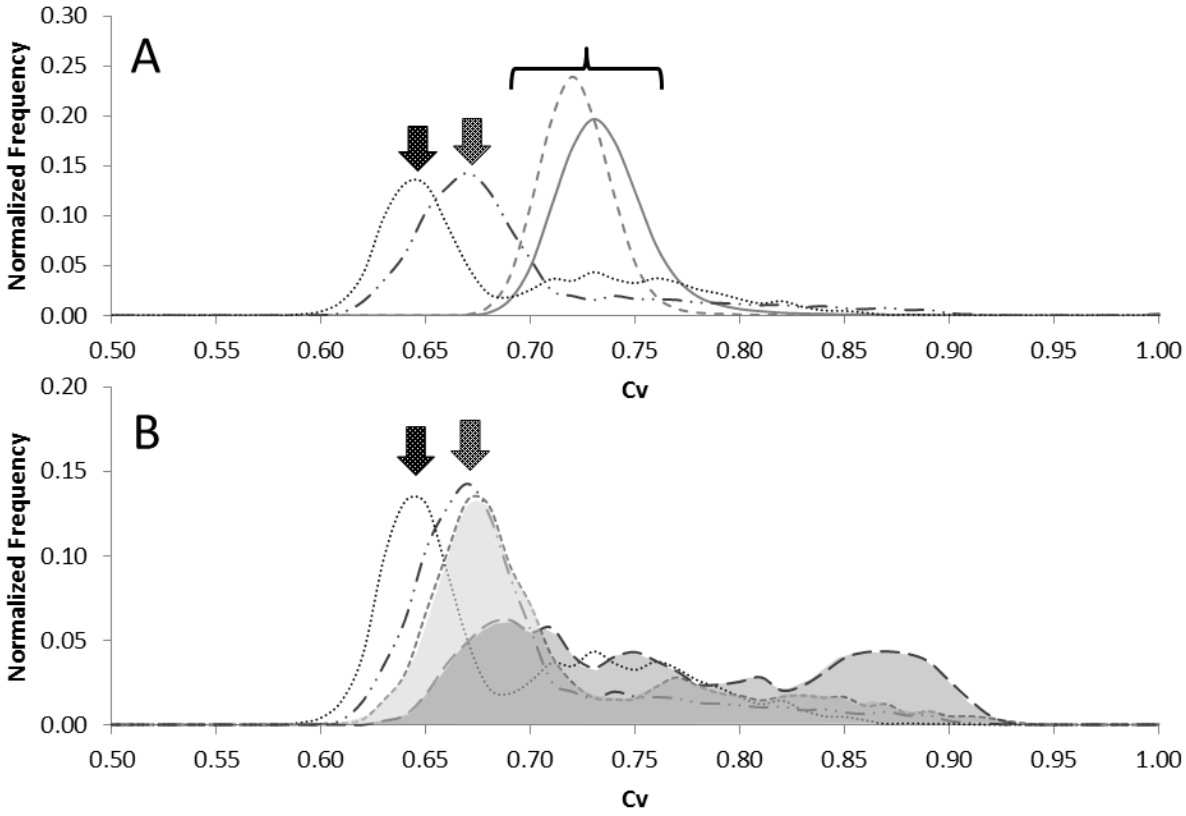


Figure 30: The C_v distributions for the trained 10×10 SOMs generated from the same parameters for the C++ and MATLAB programs. (A) Shows the rectangular geometry maps. This includes the SOM generated from C++ and the rectangular MATLAB SOMs. (B) Shows the SOMs generated by MATLAB. This includes the rectangular and hexagonal SOMs. The rectangular SOMs generated from MATLAB are seen in both A and B; the arrows show the peaks in both graphs. illustrates the peak of the bordered rectangular SOM from the MATLAB program and is the peak of the toroidal rectangular SOM from the MATLAB program. illustrates the peak of both bordered and toroidal rectangular SOMs from the C++ program. (—) bordered rectangular SOM from the C++ program (Table 7:7-ff), (·····) toroidal rectangular SOM from the C++ program (Table 7:7-ff), (– – –) bordered rectangular SOM from the MATLAB program (Table 7:7-ii), (– · –) toroidal rectangular SOM from the MATLAB program (Table 7:7-ii), bordered hexagonal SOM from the MATLAB program (Table 7:7-jj), toroidal hexagonal SOM from the MATLAB program (Table 7:7-jj).

The MATLAB program is better for generating reproducible partitionings than the C++ program for SOMs with a mapsize greater than 5×5. Unlike the C++ program, the MATLAB program could ascertain a difference in the partitioning of data by different boundary conditions (bordered and toroidal). The toroidal SOMs generated from MATLAB have higher C_v values than the bordered SOMs. The hexagonal SOMs produced a broad C_v distribution with more maps in the higher C_v region than the rectangular SOMs.

7.3.5.2 Comparing Nodal Geometry and Boundary Conditions

The goal is to compare the different SOMs and to determine which map conditions produce reproducible data partitioning into high quality clusters. The C_v distributions were compared between the rectangular and hexagonal 10×10 SOMs generated from the MATLAB program (Figure 31). The rectangular and hexagonal SOMs when compared to the same parameter SOM generated the C_v distributions in Figure 30B, where the majority complement of C_v comparisons for both boundary conditions for the rectangular and the bordered hexagonal SOMs overlap. The most dissimilar SOMs are those with the lowest C_v values: the bordered rectangular SOM (Table 7:7-ii) compared to the toroidal hexagonal SOM (Table 7:7-jj). Referring to Figure 30B, the majority complement of C_v comparisons for each of these two types of maps are the furthest apart and therefore the maps have the most dissimilar clustering partition of the dataset. The comparison of bordered rectangular SOMs (Table 7:7-ii) compared to the bordered hexagonal SOMs (Table 7:7-jj) and toroidal rectangular SOMs (Table 7:7-ii) compared to the bordered hexagonal SOMs (Table 7:7-jj) have overlapping C_v distributions in Figure 31. This overlapping region shows that the toroidal SOMs with different geometry are more similar to each other than compared to bordered SOMs with different geometry. This is probably due to the edge effect where the different geometry is affecting the partitioning. When a toroidal SOM is compared to a bordered SOM with different geometry, the C_v distributions are between the bordered with different geometry and the toroidal with different geometry. This ordering is as expected from Figure 30B where the bordered maps have a lower majority complement of the C_v comparisons than the toroidal maps. The most similar SOMs are those with the highest C_v values: the toroidal rectangular SOMs (Table 7:7-ii) compared to the toroidal hexagonal SOMs (Table 7:7-jj). However, none of these comparisons in Figure 31 produced C_v values in the upper

limit of ~0.9, which means than different data partitioning was found for different boundaries (bordered and toroidal) and different map nodal geometries (rectangular and hexagonal). However, it would suggest that toroidal boundaries tend to produce more similar clusters regardless of nodal geometry.

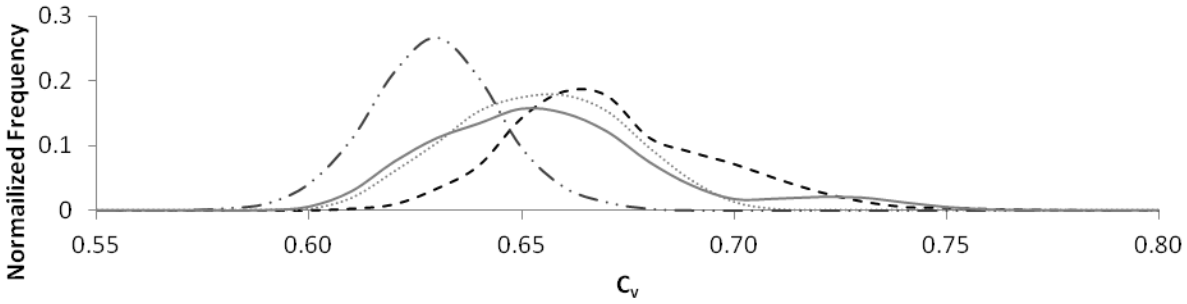


Figure 31: The normalized C_v comparison between different 10×10 nodal geometry SOMs generated from the MATLAB programs. There are 100 SOMs for each boundary and geometry condition for a total of 10000 comparisons. (——)10×10 bordered rectangular SOM (Table 7:7-ii) compared to the 10×10 bordered hexagonal SOM (Table 7:7-jj), (– – –)10×10 bordered rectangular SOM (Table 7:7-ii) compared to the 10×10 toroidal hexagonal SOM (Table 7:7-jj), (– – –)10×10 toroidal rectangular SOM (Table 7:7-ii) compared to the 10×10 bordered hexagonal SOM (Table 7:7-jj), and (……)10×10 toroidal rectangular SOM (Table 7:7-ii) compared to the 10×10 toroidal hexagonal SOM (Table 7:7-jj).

7.3.6 BATCH Trained Maps

Batch training refers to the process of updating nodal centres after all conformations are re-assigned to their BMU node (Section 2.4.1). For the sequential algorithm, the values of node centres are updated after each (randomly selected) conformation is re-assigned to its BMU (Section 2.4.2). This means that for the same training length, the sequentially trained SOMs are updated approximately N more times, where N is the total number of conformations being clustered on the SOMs. The batch trained SOMs do not possess a single majority complement C_v comparison region in their C_v distributions. Figure 32 shows a duplicate of the results for 5×5 SOMs generated from the MATLAB program shown in Figure 29, as well as those for the corresponding batch trained SOMs. For both toroidal and bordered batch trained SOMs there is

narrow peak with a C_v value of 1.00. The batch training algorithm produced identical SOMs in different runs. The maps that were different produced a majority complement of C_v comparison region (C_v ~0.50-0.75) which contained multiple peaks and valleys. This means there was less variation in the SOMs. To obtain clustering which produces a smoother C_v comparison region, the training length of the batch trained maps should be increased. The bordered sequentially trained SOMs had a similar location to the C_v distribution peaks to the bordered batch trained SOMs. However, since both trained SOMs with bordered boundaries have lower C_v values than toroidal boundaries, the toroidal SOMs have a more similar partitioning to each other compared to the bordered SOMs. The toroidal sequentially trained SOMs have a greater amount of higher C_v values than the toroidal batch trained SOMs. The sequentially trained maps are updated more frequently; this could result in the higher C_v values seen in the toroidal sequentially trained SOMs compared to the toroidal batch trained SOMs. Since, there are a greater number of maps which are reproducible in the toroidal sequentially trained SOMs, throughout the rest of this work the SOMs will only be trained with the sequential algorithm.

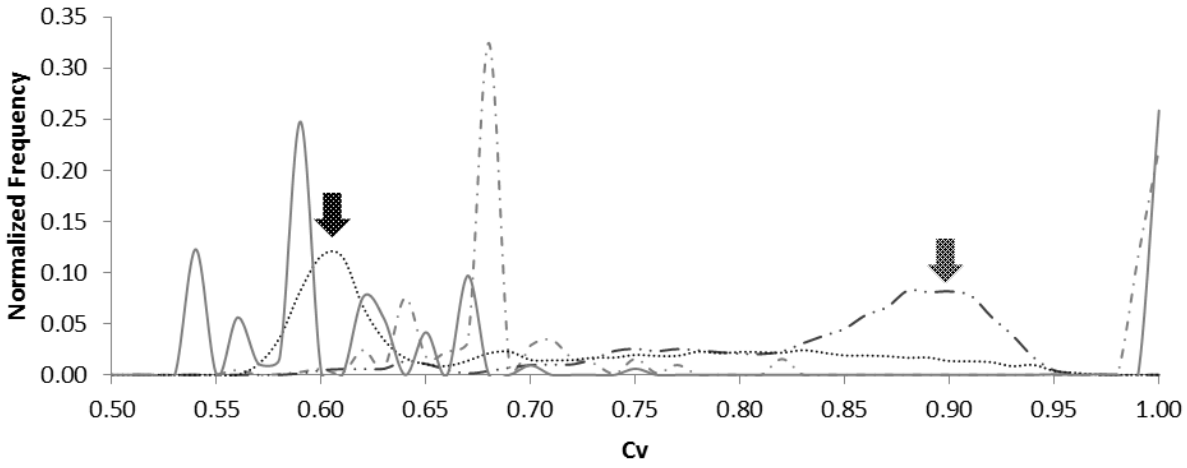


Figure 32: The C_v distributions for the sequentially and batch trained MATLAB 5×5 SOMs trained using the same parameters. The sequentially trained SOMs are duplicates of only the MATLAB data from the 5×5 graph in Figure 29. ⬇ illustrates the majority complement of C_v comparisons of the bordered rectangular sequentially trained SOMs from the MATLAB program and ⬇ is the majority complement of C_v comparisons of the toroidal rectangular sequentially trained SOMs from the MATLAB program. (……) bordered rectangular SOMs sequentially trained from the MATLAB program (Table 7:7-gg), (— —) toroidal rectangular SOMs sequentially trained from the MATLAB program (Table 7:7-gg), (——) bordered rectangular SOMs batch trained from the MATLAB program (Table 7:7-kk), and (– – –) toroidal rectangular SOMs batch trained from the MATLAB program (Table 7:7-kk).

7.3.7 Learning Rate Factor

There are three different learning rate factor algorithms in the MATLAB program: inverted, linear and power (Section 2.4.2.1). The learning rate factor helps to control the contribution to the nodal centres, this value decreases as the training length increases. The default settings in MATLAB uses an α_0 of 0.5 and the learning rate factor is the INV (equation [15]). To determine which learning rate factor best clusters the conformations of MET, 100 SOMs were trained with each of the three possible learning rate factors. The SOMs were 10×10 hexagonal toroidal SOMs with a training length of 100 iterations; the dataset used to train the SOMs was TIP3P/MD1 (Table 7:7-jj, 7-ll and 7-mm). Figure 33 shows both the Objective function (INSSQ) distribution (Figure 33 A) and the C_v distribution (Figure 33 B). The C_v distribution for SOMs produced using the learning rate factor of INV is a repeat of that in Figure 30B for the 10×10 hexagonal toroidal SOM, where smaller values of the objective function

INSSQ represent tighter clusters. The INV learning rate factor produces larger values of the objective function when compared to those from SOMs trained using both the POWER and LINEAR. The algorithms POWER and LINEAR produce tighter clusters of the dataset more frequently than INV. The INV SOMs produce a range of C_v values while the POWER and LINEAR learning rate factor produce a peak at ~0.68. Though the C_v values for INV are higher than POWER and LINEAR the higher INSSQ means the clusters produced from INV are not tight. The INV learning rate factor is excluded as a learning rate factor, Murtola *et al.* also excluded this algorithm since it gave poor topological results (sorted the data poorly on a 2D-grid) compared to the other two algorithms,[37] SOMs not only cluster based on the INSSQ but also have a neighbourhood component which means it organizes the data on a 2D-grid.

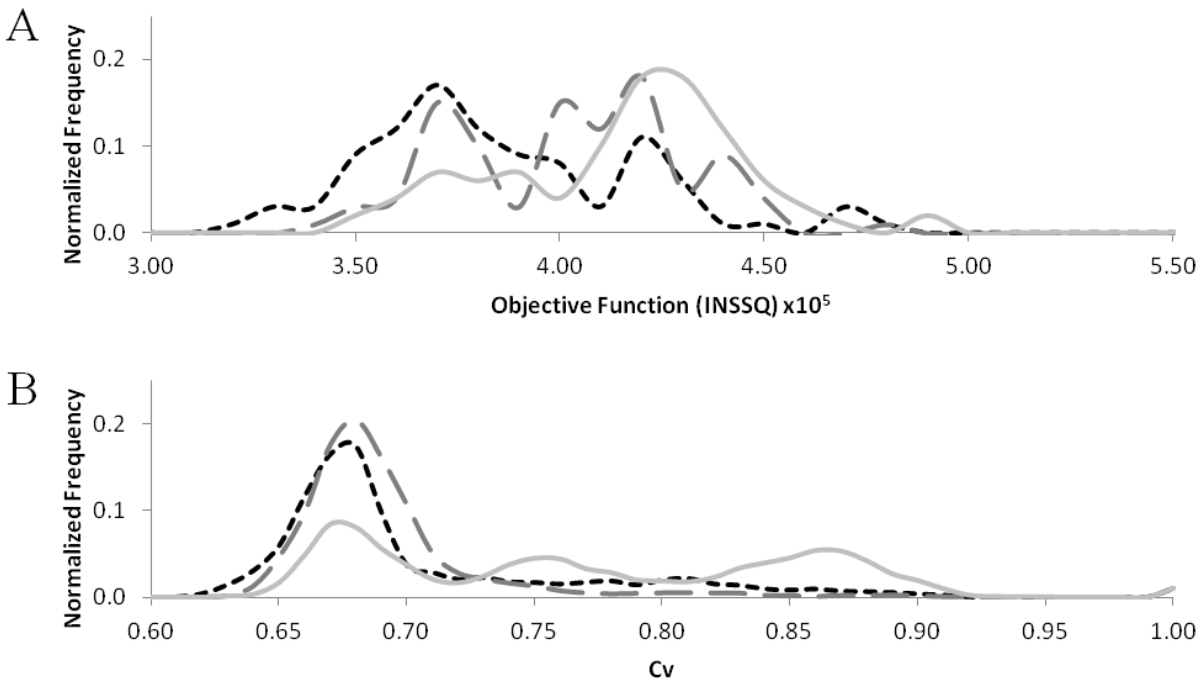


Figure 33: The C_v distributions for both Objective Function (INSSQ) and C_v for the three learning rate factors: (—) INV (Table 7:7-jj), (— —) LINEAR (Table 7:7-ll), and (....) POWER (Table 7:7-mm). The inital learning rate factor is 0.5.

Some SOMs have what is called a rough training phase phase, where the $\alpha_0$value starts at a larger value, followed by a fine tuning training where α_0 is decreased after so many iterations to a smaller value;[37] MATLAB has this option. In this case, the SOMs described above (generated with α_0 = 0.5) were used as starting points for a fine tuning phase where the α_0 was decreased from 0.05 and the learning rate factor algorithm was either LINEAR or POWER. Figure 34 shows the results after fine tuning the SOMs in Figure 33, using POWER (Figure 34A and Figure 34B) and LINEAR (Figure 34C and Figure 34D). When the fine tuning phase is done, the POWER algorithm results in objective function distributions having a peak around ~4.0-4.5×10^5, whereas the LINEAR algorithm produces maps with two distinct peaks in INSSQ. Futhermore, the C_v distributions produced after the POWER fine tuning phase are broad where the C_v distributions from the LINEAR fine tuning phase are narrow and fall at lower C_v values. Although the POWER algorithm has higher C_v values (which is desirable), the LINEAR algorithm gave more consistent results as seen in the overlap of the C_v distributions, as well as the LINEAR algorithm produced more SOMs with two distinct peaks. This will be advantageous for seeing reproducible clusters.

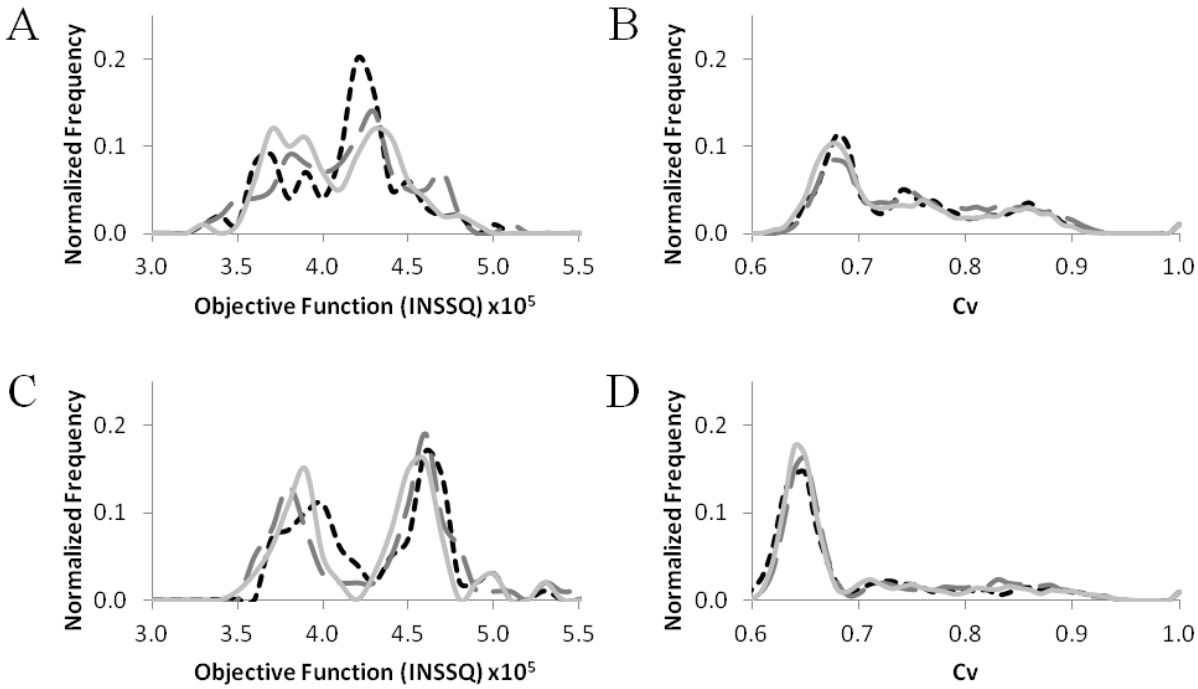


Figure 34: Fine tuning the SOMs from Figure 33, either by a power learning rate algorithm (A and B) (Table 7:7-qq, 7-rr, and 7-ss) or linear learning rate algorithm (C and D) (Table 7:7-nn, 7-oo, and 7-pp). The initial learning rate factor for the second phase was 0.05. (A and C) are distributions for the Objective Function (INSSQ) and (B and D) are C_v distributions. The labeling of the graphs is from the rough tuning phase where (—) was initialy trained with a inv algorithm (Table 7:7-qq and 7-nn), (—) was initialy trained with a linear algorithm (Table 7:7-rr and 7-oo), and (....) was initialy trained with a power algorithm power (Table 7:7-ss and 7-pp).

The LINEAR fine tuning phase produced very similar distribution of SOMs, no matter which SOM parameters are used in the rough training phase as evaluated by the overlap in both the C_v distributions and the objective function distributions. Employing the two different training phases has no advantage since the LINEAR algorithm can be used starting at the outset with an α_0 of 0.05 and this produces the same results as a SOM trained in two phases, where the second phase is using a linear algorithm with a 0.05 α_0 value (Figure 35, where Figure 34C and Figure 34D are duplicated). The optimal parameters for the learning rate factor use the LINEAR learning rate factor with a single training phase and an initial learning rate factor of 0.05. These parameters are used throughout the rest of this work.

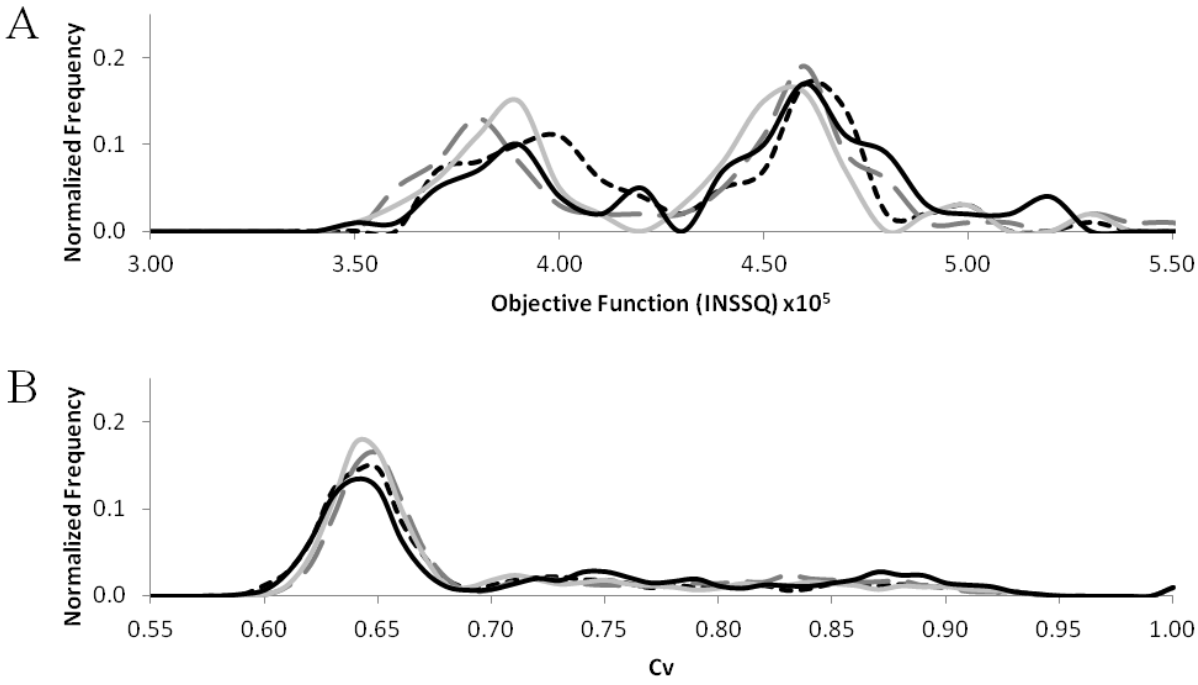


Figure 35: A duplicate of Figure 34 C and D where the fine tuning phase was a LINEAR learning rate algorithm and the graphs are labeled from there rough tuning phase where (—) was initaly trained with a INV algorithm (Table 7:7-nn), (—) was initaly trained with a LINEAR algorithm (Table 7:7-oo), (....) was initaly trained with a POWER algorithm POWER(Table 7:7-pp), and (—) are SOMs trained by the LINEAR learning rate algorithm with an inital learning rate factor of 0.05 (Table 7:7-x).

7.3.8 Training Length

The SOMs that are produced from the various MET conformational datasets should have similar C_v distributions, since all the datasets overlap by ~90% (Section 7.3.2.2.2: Table 9). The C_v distribution of the TIP3P/MD1 dataset using 100 SOMs trained with a LINEAR learning rate algorithm and an initial learning rate factor of 0.05 (Table 7:7-x) is the same in Figure 36 as that depicted in Figure 35 B. Figure 36 shows the C_v distribution generated from SOMs produced for the Classical MD simulations (Table 7:7-x and 7-y) have higher C_v values than those generated from the REMD simulations (Table 7:7-aa, and 7-cc). SOMs produced from the collected conformational sample of TIP3P/MD2 have no well-defined majority complement region in the C_v distribution while those from TIP3P/MD1, TIP3P/REMD1, and TIP3P/REMD2 have right tail skews of their majority complement region. The variations in the distributions in Figure 36

suggest that the SOMs had not been trained long enough or the SOMs could be different because the frequency of like conformations explored during different simulations might be different. Table 9 does not disclose if nodal occupancies are similar.

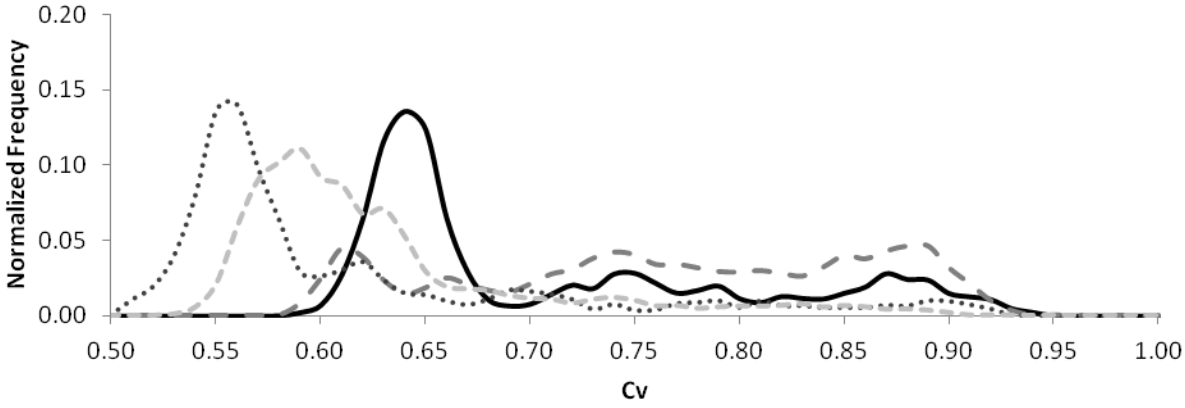


Figure 36: The C_v distributions for the optimized learning rate parameters for the MATLAB program. These 100 10×10 hexagonal SOMs (Table 7:7-x, 7-y, 7-aa, and 7-cc) were produced from each dataset. (—) is TIP3P/MD1, (–) is TIP3P/MD2, (…) is TIP3P/REMD1 and (– –) is TIP3P/REMD2.

To validate our interpretation that the training length T was insufficient, 100 SOMs were generated for each different training length: 100, 500, 1000, 5000, and 10000. Figure 37 shows the C_v distibutions for the multiple training lengths of the dataset TIP3P/MD1. As the training length increases, the majority complement region shifts to higher C_v values and a peak starts to form in the higher C_v (~0.95) region. Furthermore, the distances between the peaks in C_v distributions between these two regions starts to decrease. Recall that a favourable C_v distribution similar to Figure 27 is desirable, which shows reproducibility. For that MET sample, the optimized training length for a 10×10 hexagonal toroidal SOM (TIP3P/MD1) generated in MATLAB with linear learning rate factor of 0.05 is 5000, since C_v distributions for T = 5000 and 10000 overlap fairly well. The C++ maps were never optimized for training length, however we assume that for the SOM algorithm the number of iterations would have to be increased as well;

the C_v distribution for the optimized SOM would probably look similar to Figure 27 for the 10×10 SOM.

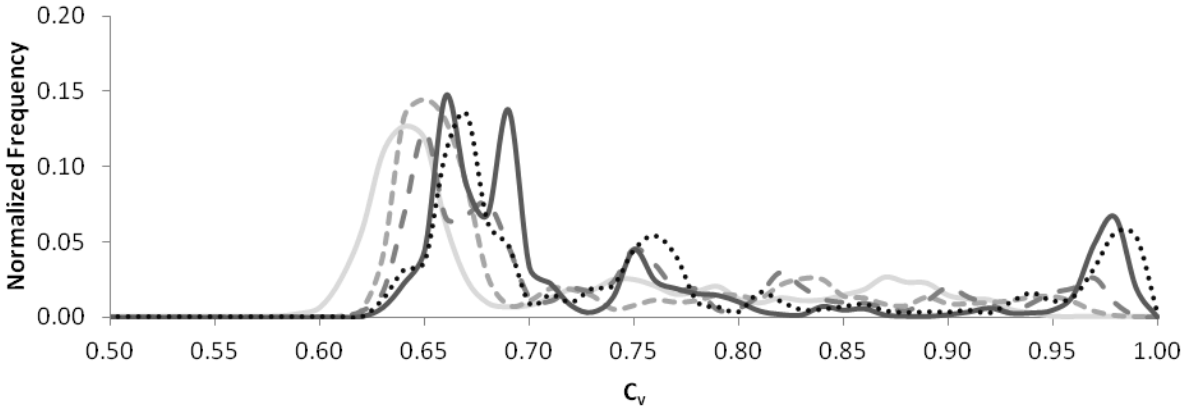


Figure 37: The C_v distributions for different training lengths of the TIP3P/MD1 dataset for the optimized learning rate parameters of the MATLAB program. These distributions are produced from 100 SOMs trained with different training lengths: (—) 100 (Table 7:7-ll), (– –) 500 (Table 7:7-tt), (–) 1000 (Table 7:7-uu), (—) 5000 (Table 7:7-vv), and (…) 10000 (Table 7:7-ww).

A similar optimization process for training length was done on the remaining MET datasets, where the optimized value for training length was found to be 5000. Figure 38A shows the C_v distributions of the MET datasets when the training length was 5000 and Figure 38B shows the C_v distributions of the MET datasets when the training length was 100. All of the C_v distributions for SOMs produced for the various solvated MET conformational samples have a majority complement region and a higher C_v value region. The classical MD simulations have three peaks in total, where the third peak is between the majority complement region and the high C_v value. The REMD SOMs produced from samples TIP3P/REMD1 and TIP3P/REMD2 have a C_v distribution with its majority complement region shifted towards higher C_v values when compared to C_v distributions generated from SOMs trained with only a training length of 100 (Figure 38B). The C_v distributions in Figure 38A are still fairly different. This could result from different partitioning of the data, the datasets cover different conformational space, or the training parameters need to be explored more. If the SOMs are partitioned differently the C_v

values can be compared to determine how similar or how different the clustering is. Comparing the occupancies of the high C_v values with high χ^2 could give an understanding of the conformational space explored by each sample; if these others do not work the training parameters would need to be looked at.

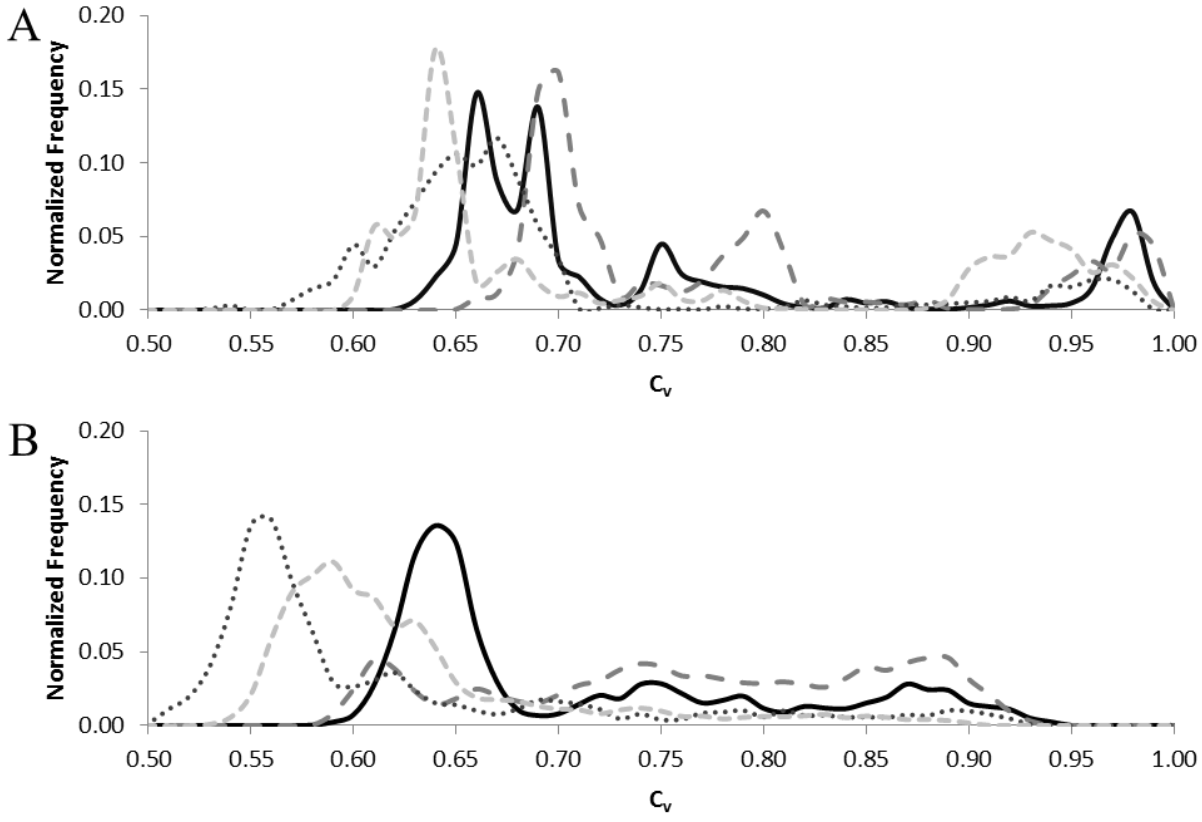


Figure 38: The C_v distributions for the optimized learning rate parameters and training length of four different datasets (A) The C_v distributions for the optimized learning rate parameters and training length of 5000 for the MATLAB program. These SOMs were produced from each dataset (—) is TIP3P/MD1 (Table 7:7-vv), (–) is TIP3P/MD2 (Table 7:7-xx), (…) is TIP3P/REMD1 (Table 7:7-yy), and (– –) is TIP3P/REMD2 (Table 7:7-zz). (B) The C_v distributions for the learning rate parameters and training length of 100 for the MATLAB program. These SOMs were produced from each dataset (—) is TIP3P/MD1 (Table 7:7-x), (–) is TIP3P/MD2 (Table 7: 7-y), (…) is TIP3P/REMD1 (Table 7: 7-aa), and (– –) is TIP3P/REMD2 (Table 7: 7-cc).

7.3.9 Reproducible Clusters in the MATLAB Program

As mentioned in section 7.3.4, SOMs could be advantageous to cluster flexible proteins, however; the lack of reproducibility means SOMs are not readily used. Figure 39 and Figure 40

are produced from 100 10×10 toroidal hexagonal SOMs that have been trained for 5000 iterations (Table 7:7-vv and 7-yy). Figure 39 and Figure 40 are similar to Figure 27 except they use a different objective function. The SOMs that represent quality clustering are likely to have low objective functions (tight clusters) and a high C_v values (reproducible). The same program was used as in Section 7.3.4 to determine the number of high quality SOMs representing different conformational partitioning. Once the SOMs were divided into reproduced groups, the SOMs were matched to their INSSQ objective functions. The groupings in Figure 39 and Figure 40 show SOMs with very similar partitionings.

Figure 39 is a scatter plot of the INSSQ objective function versus the C_v values for SOMs produced for the TIP3P/MD1 dataset. The partitioning of groups which contain three or more reproduced SOMs are shown in colour. The partitioning group with the highest occupancy has 41% of the SOMs and has a low objective function. Comparing the most reproducible 10×10 SOM from Figure 39 (41%) with that found for 5×5 SOMs in Figure 27 (36.8%) where the SOMs were generated from the same dataset (TIP3P/MD1), the values are very similar. One map was taken from the largest reproducible group of the 5×5 SOMs and another from thelargest reproducible group of the 10×10 SOMs; these SOMs were compared by C_v. The C_v value between these two maps was 0.88. This implies not only can SOMs be reproducible but it is not dependent on the source code or mapsize, since two independent source codes and different mapsizes produced similar partitions with a high C_v value. Therefore the same conformations were grouped together on these two maps.

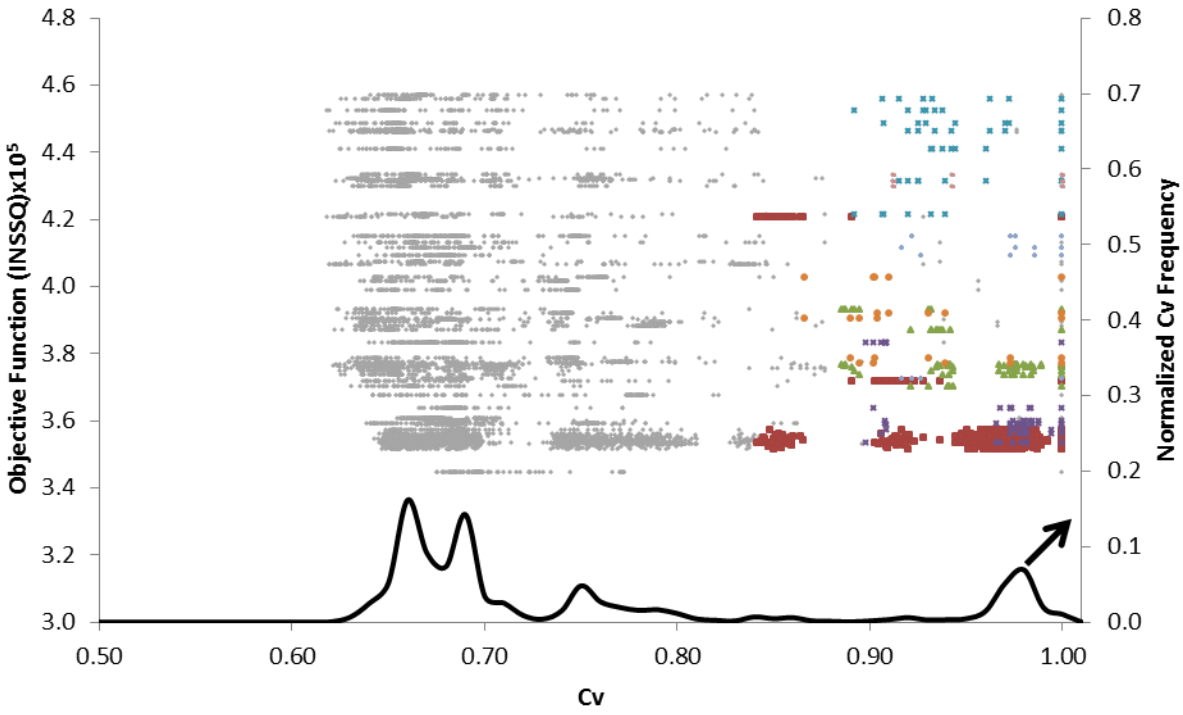


Figure 39: The INSSQ objective function versus C_v values for the 100 toroidal 10×10 hexagonal SOMs trained by the TIP3P/MD1 dataset from the MATLAB program (Table 7:7-vv). The scatter graph is the INSSQ objective function (left axis) against C_v (x-axis). When a C_v distribution is generated from the scatter graph (—), the distribution is a duplicate of the same dataset in Figure 37 and Figure 39 (normalized C_v frequency; right axis). The C_v is generated from two maps; for each map a INSSQ objective function was computed (one C_v value has two INSSQ objective functions). ◆ are all the C_v values, the coloured points are the 7 groups of SOMs which produce high C_v values with each other and have 3 or more maps in a group. ■ is the highest occupied group with 41% of the SOMs in this group.

Figure 40 is a scatter plot of the INSSQ objective function versus the C_v value from a different dataset (TIP3P/REMD1) than shown in Figure 27 and Figure 39. The highest occupancy group (23%) is not one which has the lowest value of the objective function; however, the program found groupings that are the second (17%) and fourth (15%) highest occupancy which do have lower values of the objective function. These other groupings with lower objective function values could represent a better partitioning than the highest occupancy group. The multiple groups with a range of objective functions could mean that the clustering is finding different geometric groupings of the data. The criteria could be more dominant in the

TIP3P/REMD1 dataset than the TIP3P/MD1, since TIP3P/REMD1 could cover more conformational space than TIP3P/MD1.

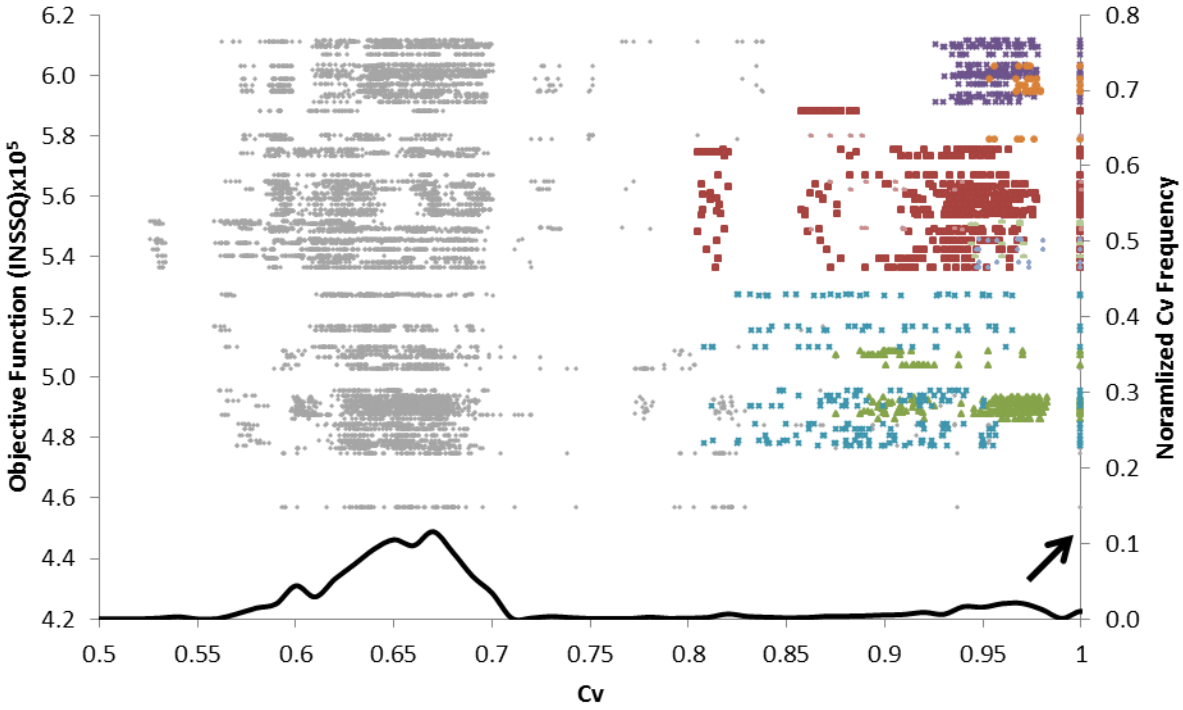


Figure 40: The INSSQ objective function versus C_v values for the 100 toroidal 10×10 hexagonal SOMs trained by the TIP3P/REMD1 dataset from the MATLAB program (Table 7:7-yy). The scatter graph is the INSSQ objective function (left axis) against C_v (x-axis). When a C_v distribution is generated from the scatter graph (—), the distribution is a duplicate of the same dataset in Figure 38 (normalized C_v frequency; right axis). The C_v is generated from two maps; for each map a INSSQ objective function was computed (one C_v value has two INSSQ objective functions). ◆ are all the C_v values, the coloured points are the 8 groups of SOMs which produce high C_v values with each other and have 3 or more maps in a group. ■ is the highest occupied group with 23% of the SOMs in this group. ▲ is the second highest occupancy with 17%. ✸ is the fourth highest occupancy of 15%.

7.4 Conclusion

The use of C_v and rSI to evaluate reproducibility of SOMs produced using conformational datasets of MET outperformed the SI comparison. However, the C_v value was chosen as most useful due to its faster computation time and its ability to compare nodal memberships via χ^2 values. In order to be confident that SOM partitioning is reproducible, SOMs must be run multiple times (~100) and a method of comparing them should be employed. The C_v distribution was able to compare SOMs for independently generated SOMs as long as

they were produced from partitioning the same dataset. When conformational ensembles must be compared, supervised training by a previously trained SOM of another dataset can be used. This enables the use of C_v, since C_v can only be used when comparing maps with the same dataset on the SOM. The optimal values for training of SOMs for the number of training phases (one), training algorithm (sequential), neighbourhood (a form of decreasing depending on the program), map boundaries (toroidal), learning rate algorithm (LINEAR) and learning rate factor (0.05) were determined. It was also determined that a combination of low objective function and high C_v values must be used to parameterize the SOMs. However, further exploration should be done to determine which is more important in terms of the partitioning of the data. From the test which was performed using the 5×5 SOMs produced from the C++ program and the 10×10 SOM produced by the MATLAB program, the programs produced similar partitioning of the dataset. However, the comparison between groups formed from the different parameters (mapsize, source code, and geometry) must be further explored. The clusters produced by SOMs in this work, find the four clusters obtained by the two-dimensional plots of PCA by Sanbonmatsu and García,[59] however we disagree that there are only four predominant structures. Sanbonmatsu and García described the four structures by distances; however, many of the clusters could not be organized into the four groups.

8 Intrinsically Unstructured Proteins: human Parathyroid Hormone (hPTH)

Human Parathyroid Hormone (hPTH) is a flexible 84 amino acid hormone that is associated with the homeostasis of Ca^{2+} in the extracellular environment.[81–83] The control of Ca^{2+} concentration in the blood is regulated by a negative feedback loop to the parathyroid gland;

when Ca^{2+} is low or when the phosphate concentration is high, hPTH is produced. The hPTH interacts with PTH-receptor (PTHR), which is a G protein-coupled receptor.[84] The PTHRs are located on the surface of renal tubule cells in the kidney and osteoblasts in bone tissue. The increase of the Ca^{2+} concentration in the extracellular environment is accomplished by stimulating Ca^{2+} resorption in the kidney or by increasing osteocastic bone resorption of Ca^{2+} in the bone matrix.

The hPTH (1-34) N-terminal fragment has been shown to have endocrine biological activity.[85] This synthetic peptide has been approved by the FDA (the generic name is teriparatide) as a treatment for osteoporosis. [85] The synthetic fragment and the natural occurring ligand interact with the same PTHR. The hPTH interacts through a two stage binding process (Figure 41).[86,87] The first stage involves the C-terminal α-helix of hPTH interacting with the N-terminal extracellular domain of the PTHR; this stage activates the receptor.[88,89] The second stage is the rate limiting step, where the N-terminal α-helix of hPTH interacts with the juxtamembrane region of the receptor (extracellular loops and the seven transmembrane helices);[90] this stage causes a conformational shift in the receptor.[91] The first stage of the binding has been detailed by an X-ray structure (3C4M; hPTH (15-34)).[92]

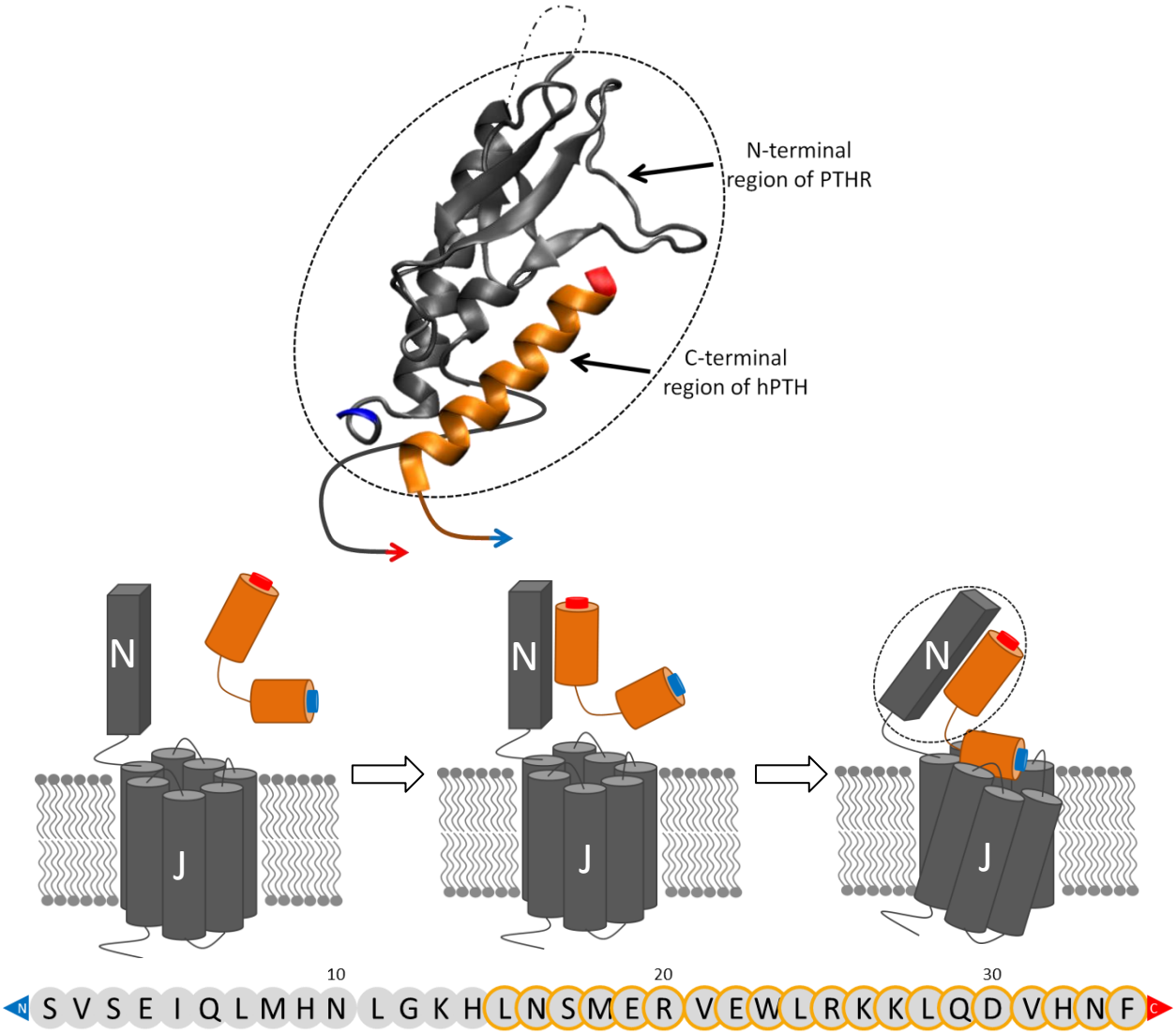


Figure 41: The two step binding process of hPTH(1-34) (orange; sequence under image) to the PTHR (grey). hPTH has two α-helices; the cylinder marked with the red is the C-terminal α-helix and the cylinder with the blue is the N-terminal α-helix; they are connected by the hinge region. The C-terminal α-helix of hPTH interacts with the N-terminal extracellular domain of the PTHR (N) then the N-terminal α-helix of hPTH interacts with the juxtamembrane region (J). The interaction of the N-terminal α-helix of hPTH causes a change in conformation of the receptor to its active form. The first stage has been characterized by X-ray crystallography (PDB: 3C4M)(oval with a dotted line shows the X-ray structure). (_ . _) missing atoms in the X-ray structure. The arrows compare the cartoon image to the X-ray structure. The blue arrow shows where the N-terminal α-helix of hPTH would connect and the red arrow shows where the juxtamembrane regionattaches to the N-terminal extracellular domain of the PTHR. The C-terminal α-helix of hPTH has been characterized in the X-ray structure for amino acids 15-34.[92]

The general consensus is that the structure of hPTH is mostly helical. The two helixes encompass the C-terminal region (~15-34) and the N-terminal region (~1-14). The helices are connected by a hinge region,[93] which is highly flexible. The C-terminal α-helix is more stable

than the N-terminal α-helix. Many of the NMR structures of hPTH show a U-shape that contains a hydrophobic core thought to be the active form.[94] The U-shape is through to be the active form to get the PTHR to change in conformation, upon binding of the N-terminal helix of hPTH.

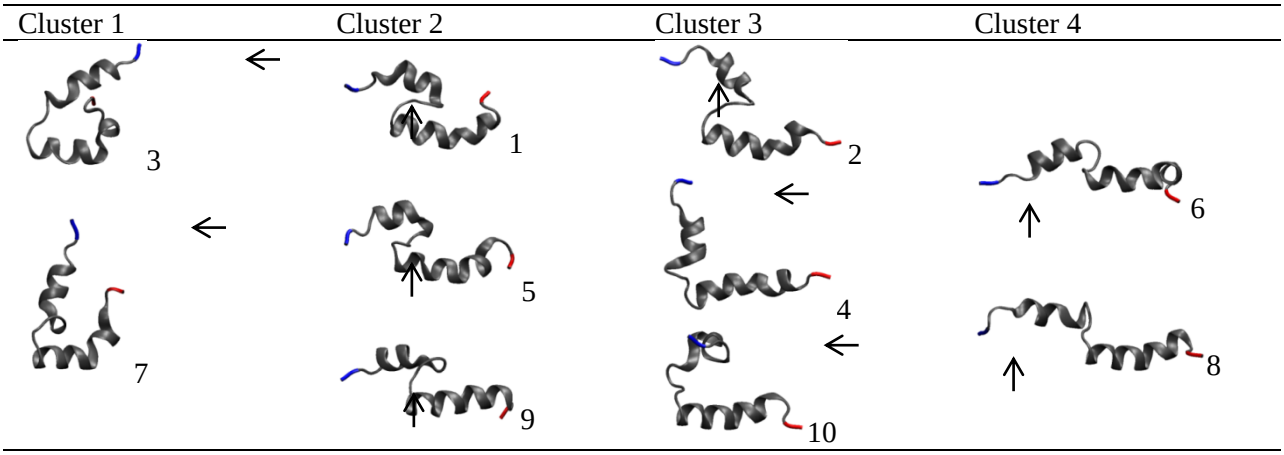
8.1 Simulations with hPTH

The fragment hPTH (1-37) has a broad conformational distribution. The broad conformational distribution can be demonstrated with visual assessment of the diversity of the multiple NMR structures in an aqueous environment. Marx *et al.* deposited some of the many NMR structures into the Protein Data Bank.[94,95] These structures were generated from restrained molecular dynamics for both hPTH(1-37)[95] and hPTH(1-34)[94]. The solved NMR structures for the two peptides of different lengths are similar to each other and the elongation of the sequence had no effect on the secondary structure.[94,95] An unrestrained MD simulation was done on hPTH(1-37) solvated by a TIP3P water model to ascertain the stability of the different helical regions. As well, it was determined that a small hydrophobic core (U-shaped) existed, corresponding to the long range NMR coupling, between the two helixs.[95] The "U-shape" was further explored by Pellegrini *et al.*; although there was no experimental evidence to validate the tertiary structure,[96] the "U-shape" is still thought to be the active form of hPTH. However, two conformations were determined from the ensemble-based calculations, both a "U-shape" and a linear shape.

The hPTH(1-34) is a therapeutic protein that is susceptible to oxidation. In order to design rational approaches for stabilizing the degradation processes of oxidation, Chu *et al.* explored the relationship between the conformational properties of hPTH(1-34) and the oxidation of methionine residues (Met8 and Met18).[97] This group performed 10 MD simulations on hPTH starting from each model of 1ZWA in PDB (Table 10).[94] These MD simulations

formed a benchmark for the appropriate sampling of conformational space of hPTH. They found that the MD simulations deviated significantly from their starting NMR structure, meaning the protein is very flexible. Furthermore, when each trajectory was compared to the original 10 NMR models the trajectories were most similar to the NMR structure from which the simulation started. They proposed a water–mediated oxidation mechanism. Met 8 had more frequent contacts with other amino acids, limiting its exposure to solvent, leading to a lower oxidation rate than that of Met18. Since Met18 is located near the bottom of the "U-shape", it had a greater exposure to solvent.

Table 10: The 10 models of hPTH in 1ZWA. These models are organized based on Trout's research on radius of gyration of their α-carbons. [97]



The arrows show the location of the N-terminal end of hPTH

Clustering of hPTH is not trivial due to its broad conformational distribution. Chu *et al.* roughly clustered the NMR structures into four groups based on their radius of gyration (Table 10); however, the MD simulations deviated significantly from their starting NMR structures.[97] An earlier study by Gordon and Somorjai used fuzzy clustering to group the conformations from different MD simulations of fragments of hPTH (1-34, 13-34, 20-34), since the exact structure of hPTH is not known in solution.[2] The MD simulations were run in implicit water with ε =72 or 2. They collected ~1000 conformations from each MD simulation, to be clustered by their root-

mean-squared-inter-$C_\propto$-distances. They determined that one to three clusters were determined for each simulation.[2] Futhermore, the MD simulations deviated from the original conformation significantly. The conformation closest to the cluster centre was used to represent the members of the whole cluster and can be used in further analysis.[2] Fuzzy clustering could find an optimal number of clusters for the different MD simulations. The simulations run at the same temperature could use all-atom or inter-α-carbon distances to cluster the conformations, both of which gave similar results. This meant the differences in conformation are determined by the backbone atoms of the protein and not the sidechains. Fuzzy clustering showed that the simulations were not run long enough to sample all of conformational space.

8.2 Methods and Materials for hPTH

8.2.1 Solvated MD Simulation

The starting conformations of the zwitterion hPTH(1-34) were taken from 1ZWA (Table 10). Alexandria DiCosimo (Brock University Science Mentorship Student 2012), assisted by Michelle A. Eisner, performed ten all-atom molecular dynamics (MD) simulations on hPTH using NAMD 2.7[48] on SHARCNET[78] in an NPT ensemble using the CHARMM c31b1 force field.[46] The protein was solvated by 9254 TIP3P[49] water molecules contained in a box with dimensions of 71×62×56 Å. The potential energy of the system was calculated using periodic boundary conditions. The cutoff distance for the non-bonded terms was 12 Å and the switching distance was 8 Å. The pair-list distance was generated between non-bonded atoms under 13.5 Å and was generated twice per cycle, where the cycle was 20 time steps. Covalently bonded hydrogens were constrained to their equilibrium bond distances using the SHAKE algorithm.[51] The potential energy of the solvated system was minimized for 1000 conjugate gradient steps to reduce the unfavourable interactions due to atom overlap between the TIP3P water and the

protein. The temperature of the system was raised by 25 K every 2 ps until 300 K was reached. Langevin dynamics[48] and Berendsen pressure bath coupling[55] were used to maintain a constant temperature of 300 K and a constant pressure of 1 atm. The Velocity Verlet algorithm was used to perform the MD simulation. An integration time step of 2 fs was used. The system was equilibrated for 16 ps. The simulation was run for a further 2 ns and the coordinates were written to trajectory files every 400 fs for a total of 5125 conformations that were used for analysis.

8.2.2 Solvated REMD Simulations

Using the coordinates after the 16 ps equilibration period starting from model 2 of 1ZWA, an REMD simulation of hPTH(1-34) was run. The REMD simulations were carried out by Stephanie Philpott (Brock University Science Mentorship Student 2013) and Emma Kennedy (Brock University Science Mentorship Student 2014), assisted by Michelle A. Eisner, using NAMD 2.9[48] on SHARCNET[78] in an NVT ensemble using the CHARMM c31b1 force field.[46] The REMD simulation had 64 replicas running in parallel at different temperatures ranging from 275-419 K: 275.00, 276.84, 278.70, 280.57, 282.45, 284.35, 286.25, 288.17, 290.11, 292.05, 294.01, 295.98, 297.97, 299.96, 301.98, 304.00, 306.04, 308.09, 310.16, 312.24, 314.33, 316.44, 318.56, 320.70, 322.85, 325.02, 327.20, 329.39, 331.60, 333.82, 336.06, 338.32, 340.58, 342.87, 345.17, 347.48, 349.81, 352.16, 354.52, 356.90, 359.29, 361.70, 364.13, 366.57, 369.03, 371.50, 373.99, 376.50, 379.03, 381.57, 384.13, 386.70, 389.30, 391.91, 394.54, 397.18, 399.85, 402.53, 405.23, 407.95, 410.68, 413.44, 416.21, and 419.00 K. Exchanges were attempted every 2 ps and coordinates were written to trajectory files every 400 fs. The average exchange ratio was 34.5%. The REMD simulations were carried out for 68 ns using an integration step of 2 fs, producing a total of 54110 conformations per replica. The conformations at 299.96 K were analysed since the temperature was the closest to the original temperature 300 K in the MD simulations.

8.2.3 hPTH Datasets

The backbone dihedral angles ϕ and ψ were extracted from the conformations of hPTH produced from the MD simulations (10×5125 conformations) and the REMD simulations (54110 conformations). The dihedral angles, other than the first and last, were transformed into metric coordinate space,[10,67] to account for the circular statistics of angular variables, producing a 132-dimensional dataset ((34 amino acids × 2 angles × 2 components) - (2 angles × 2 components) = 132 dimensions). The original datasets from the simulations are summarized in Table 11.

Table 11: Names of the original 132-dimensional datasets generated from the different simulations of hPTH

Name of Dataset	Sample	Environment	Simulation Type	Number of Conformations
hPTH/MD1	1	Solvated (TIP3P)	Classical MD	5125
hPTH/MD2	2	Solvated (TIP3P)	Classical MD	5125
hPTH/MD3	3	Solvated (TIP3P)	Classical MD	5125
hPTH/MD4	4	Solvated (TIP3P)	Classical MD	5125
hPTH/MD5	5	Solvated (TIP3P)	Classical MD	5125
hPTH/MD6	6	Solvated (TIP3P)	Classical MD	5125
hPTH/MD7	7	Solvated (TIP3P)	Classical MD	5125
hPTH/MD8	8	Solvated (TIP3P)	Classical MD	5125
hPTH/MD9	9	Solvated (TIP3P)	Classical MD	5125
hPTH/MD10	10	Solvated (TIP3P)	Classical MD	5125
hPTH/REMD	2	Solvated (TIP3P)	REMD	54110

The REMD dataset is much larger than all the others and can be broken up into smaller datasets. Furthermore, the protein hPTH(1-34) is larger than the penta-peptide and the dimensionality of the dataset can be reduced depending on the goal of clustering the conformations (Section 8.3.1). Table 12 shows the names of the new smaller datasets generated by random sampling with replacement (Section 8.3.1). These datasets contain 5000 randomly selected conformations from the original hPTH/REMD dataset.

Table 12: Names of the subsets randomly selected from original hPTH/REMD dataset

Name of the Data-subset	Original Dataset	Original Number of Conformations	Percentage of Conformations in the Data-subset	Number of Conformations in the Data-subset
REMD_hPTH1	hPTH/REMD	54110	9.24%	5000
REMD_hPTH2	hPTH/REMD	54110	9.24%	5000
REMD_hPTH3	hPTH/REMD	54110	9.24%	5000
REMD_hPTH4	hPTH/REMD	54110	9.24%	5000
REMD_hPTH5	hPTH/REMD	54110	9.24%	5000

8.2.4 SOM Analysis on hPTH Datasets

The different SOMs were produced using a number of different parameters. Table 13 is a list of parameters associated with all the SOMs discussed in this chapter. The parameters are labeled 8-a to 8-qq, where the number is the heading number and the letter(s) is associated with a particular SOM. The SOM parameters in Table 13 are presented in order of appearance in the chapter. When dealing with untrained maps there are unused training parameters; these are represented by (---).

Table 13: The parameters used to make toroidal SOMs with the hPTH datasets

Para	Dataset/Data-subsets	Program	$\eta \times \eta$	Nodal Geom[a]	Initial[b]	Train[c]	η_τ[c]	$\alpha(\tau)$[d]	T[c]	# of SOMs
8-a	REMD_hPTH1	C++	6×6	rect	maxmin	---	---	---	0	100
8-b	REMD_hPTH2	C++	6×6	rect	maxmin	---	---	---	0	100
8-c	REMD_hPTH3	C++	6×6	rect	maxmin	---	---	---	0	100
8-d	REMD_hPTH4	C++	6×6	rect	maxmin	---	---	---	0	100
8-e	REMD_hPTH5	C++	6×6	rect	maxmin	---	---	---	0	100
8-f	REMD_hPTH1	MATLAB	6×6	rect	maxmin	---	---	---	0	100
8-g	REMD_hPTH2	MATLAB	6×6	rect	maxmin	---	---	---	0	100
8-h	REMD_hPTH3	MATLAB	6×6	rect	maxmin	---	---	---	0	100
8-i	REMD_hPTH4	MATLAB	6×6	rect	maxmin	---	---	---	0	100
8-j	REMD_hPTH5	MATLAB	6×6	rect	maxmin	---	---	---	0	100
8-k	REMD_hPTH1	MATLAB	6×6	hexa	maxmin	---	---	---	0	100
8-l	REMD_hPTH2	MATLAB	6×6	hexa	maxmin	---	---	---	0	100
8-m	REMD_hPTH3	MATLAB	6×6	hexa	maxmin	---	---	---	0	100
8-n	REMD_hPTH4	MATLAB	6×6	hexa	maxmin	---	---	---	0	100
8-o	REMD_hPTH5	MATLAB	6×6	hexa	maxmin	---	---	---	0	100
8-p	REMD_hPTH1	C++	6×6	rect	maxmin	Seq	[9]	[14] [ii]	100	100
8-q	REMD_hPTH1	C++	6×6	rect	maxmin	Seq	[9]	[14] [ii]	500	100
8-r	REMD_hPTH1	C++	6×6	rect	maxmin	Seq	[9]	[14] [ii]	1000	100
8-s	REMD_hPTH1	C++	6×6	rect	maxmin	Seq	[9]	[14] [ii]	5000	100
8-t	REMD_hPTH1	C++	6×6	rect	maxmin	Seq	[9]	[14] [ii]	10000	100
8-u	REMD_hPTH1	C++	6×6	rect	maxmin	Seq	[9]	[14] [ii]	15000	100
8-v	REMD_hPTH1	MATLAB	6×6	hexa	maxmin	Seq	[10]	[16] [i]	100	100
8-w	REMD_hPTH1	MATLAB	6×6	hexa	maxmin	Seq	[10]	[16] [i]	500	100
8-x	REMD_hPTH1	MATLAB	6×6	hexa	maxmin	Seq	[10]	[16] [i]	1000	100
8-y	REMD_hPTH1	MATLAB	6×6	hexa	maxmin	Seq	[10]	[16] [i]	5000	100
8-z	REMD_hPTH1	MATLAB	6×6	hexa	maxmin	Seq	[10]	[16] [i]	10000	100
8-aa	REMD_hPTH1	MATLAB	6×6	hexa	maxmin	Seq	[10]	[16] [i]	15000	100
8-bb	REMD_hPTH1	MATLAB	6×6	hexa	maxmin	Seq	[10]	[16] [i]	20000	100
8-cc	REMD_hPTH4	C++	10×10	rect	maxmin	Seq	[9]	[14] [ii]	500	100
8-dd	REMD_hPTH4	C++	10×10	rect	maxmin	Seq	[9]	[14] [ii]	1000	100
8-ee	REMD_hPTH4	C++	10×10	rect	maxmin	Seq	[9]	[14] [ii]	5000	100
8-ff	REMD_hPTH4	C++	10×10	rect	maxmin	Seq	[9]	[14] [ii]	10000	100
8-gg	REMD_hPTH4	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [i]	5000	100
8-hh	REMD_hPTH4	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [i]	10000	100
8-ii	REMD_hPTH4	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [i]	15000	100
8-jj	REMD_hPTH4	MATLAB	10×10	hexa	maxmin	Seq	[10]	[16] [i]	20000	100
8-kk	REMD_hPTH2	C++	6×6	rect	maxmin	Seq	[9]	[14] [ii]	10000	100
8-ll	REMD_hPTH3	C++	6×6	rect	maxmin	Seq	[9]	[14] [ii]	10000	100
8-mm	REMD_hPTH4	C++	6×6	rect	maxmin	Seq	[9]	[14] [ii]	10000	100
8-nn	REMD_hPTH5	C++	6×6	rect	maxmin	Seq	[9]	[14] [ii]	10000	100
8-oo	REMD_hPTH4	C++	10×10	rect	maxmin	---	---	---	0	100
8-pp	REMD_hPTH4	MATLAB	6×6	hexa	maxmin	Seq	[10]	[16] [i]	15000	100
8-qq	REMD_hPTH4	MATLAB	6×6	rect	maxmin	Seq	[10]	[16] [i]	15000	100

[a] description can be found in section 2.1
[b] description can be found in section 2.3
[c] description can be found in section 2.4 square brackets refer to the equation numbers
[d] description can be found in section 2.4.2.1 square brackets refer to the equation numbers
--- denoted training parameters which are not used
[i] $\alpha_0 = 0.05$
[ii] $\alpha_0 = 0.1$

8.3 Results and Discussion hPTH

8.3.1 dPCA Analysis of hPTH/REMD dataset

The dihedral principle component analysis (dPCA), in the program *carma*[*],[65,66] was used to determine the regions in the protein that are the most conformationally variable in the hPTH/REMD dataset. The variable regions are dihedral angles in the protein that "move" more than others during the simulation. Using the eigenvalues and their contributions the total sample variance, it can be determined how many principal component (PC) describe 80% of the variance. In hPTH, 35 PCs are used to describe 80% of the variance. Figure 42 is a plot of the cumulative percent variance by the number of PCs. The Figure 42 shows the 35 PC to be the cutoff to describe 80% of the variation in the dataset.

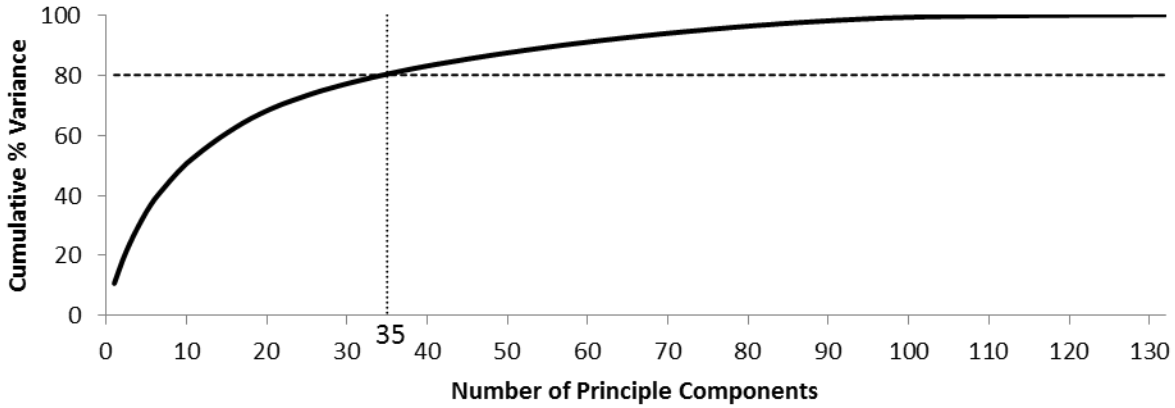


Figure 42: The percentage of compounded eigenvalues over the 132 principle components. The lines show 80% variance in the dataset (▬); the sum of eigenvalues up to and including principle component 35 describes 80% of the variance (....).

To determine the variation in the dataset contributed by each individual amino acid, the eigenvalues corresponding to 80% variance were used in conjunction with the squared coefficients of the eigenvectors. The squared coefficients were summed for all components for

[*] The program caram is a free program made by Nicholas M. Glykos.[66] The version 1.2 was used to perform the dPCA.

one amino acid (one amino acids has four components when not a terminal amino acid; i.e. two transformed values for each φ and ψ (equation [50])). Figure 43 shows all 34 amino acids and their sum of squared coefficients for each of the 35 PCs (amino acids 1 and 34 have two fewer eigenvectors). There are 35 bars contained in one amino acid column, one for each PC. The left most histogram bar (black) represents the extent to which the dihedral angles for each amino acid contributes to the first PC describing 10.7% of the sample variance. The right most bar (light grey) represents the extent to which the dihedral angles for each amino acid contributes to the thirty fifth PC describing 0.6% of the sample variance. The C-terminal helix, residues 18-28 has a lower sample variance. The terminal ends of the protein (residues 1-4 and 32-34) show large contributions to conformational variability, (Figure 43, lower) as evidenced by larger histogram bars. When the terminal ends were excluded (Figure 43, upper), this means the hinge region (residues 10-17) has the most conformational variability in the sample. For the purpose of exploring the SOMs, the terminal ends will be excluded. Figure 43 lower shows the entire amino acids 5-31 were used to generate the datasets used to train the SOMs. This analysis was used to exclude transformed dihedral angles and the dPCA variables were not used to train SOMs. The exclusion of the terminal ends changes the dataset from a 132 to a 108-dimensional dataset.

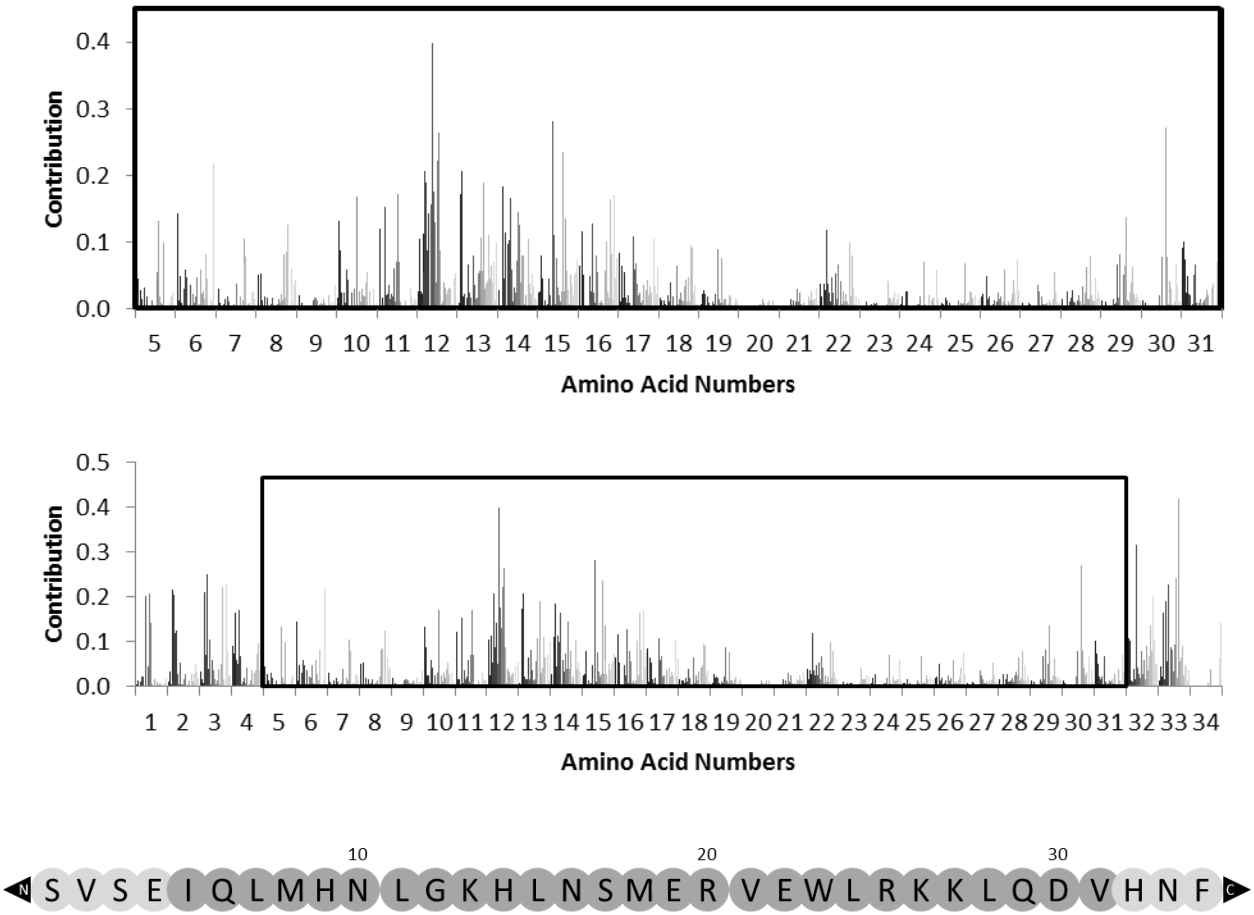


Figure 43: The contribution of transformed (ϕ, ψ) dihedral angles to eigenvector (PC) of each amino acid described for the 35 PCs that describes 80% of the variance. The enlarged figure shows amino acids 5-31, excluding the high variance terminal regions. The sequence is shown below highlighting amino acids 5-31.

8.3.2 Random Sampling with Replacement

The hPTH/REMD dataset is a much larger dataset than the ones used with MET; because hPTH has more amino acids than MET. This means the hPTH dataset has more dimensions, as well as more conformations than the MET datasets. The number of conformations can be decreased by generating a data-subset. This data-subset can be generated by random sampling with replacement. Random conformations can be taken out of the original dataset (hPTH/REMD). These conformations can be duplicates, since the conformations being taken out are placed back for the next random pick (sampling with replacement). The goal of this type of

sampling is to generate smaller datasets (data-subsets) that reflect the larger dataset. The data-subsets must be evaluated to determine if they are representative of the original dataset.

Figure 44 demonstrates that the data-subsets of 5000 conformations were drawn uniformly from the dataset hPTH/REMD (54110 conformations). The 54110 conformations from the original hPTH/REMD dataset are divided into 10 sequential sections, as shown across the x-axis of Figure 44. The proportion of the 5000 randomly selected members drawn from each of these 10 labelled sections was computed for each of the five subsets. If the sampling with replacement was done correctly, on average, 10% of the 5000 members of the subset should come from each of the 10 labelled regions. Figure 44 shows that with one standard deviation, this is true in most cases and certainly is true within two standard deviations.

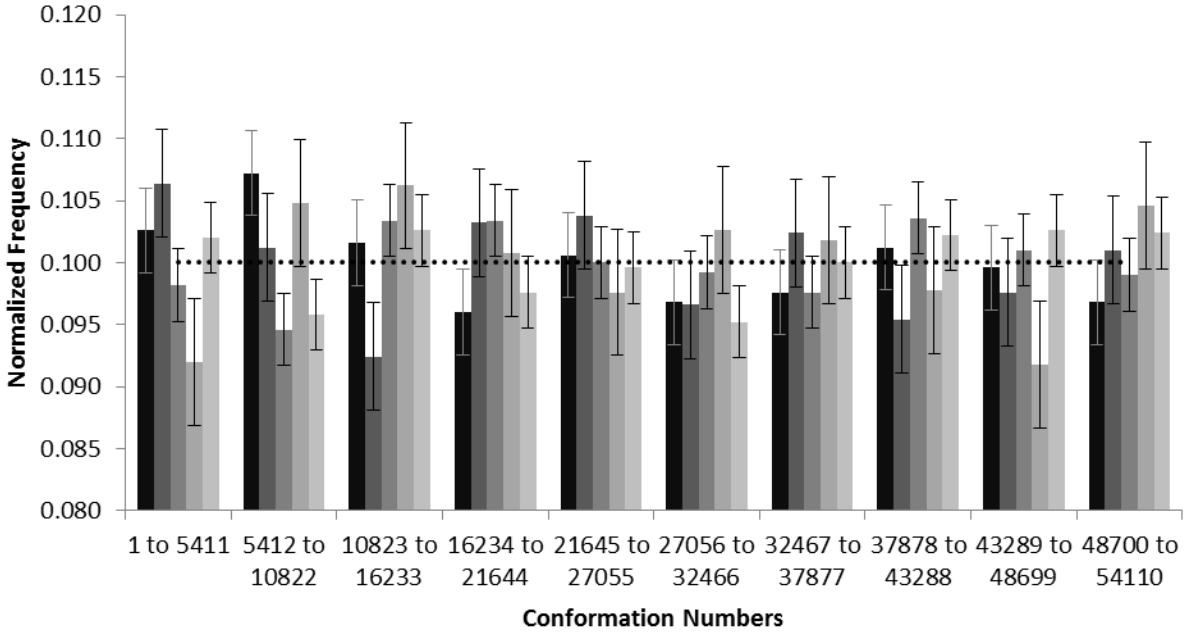


Figure 44: Proportion of 5000 member data-subsets drawn by sampling with replacement from 54110 hPTH/REMD conformations. The error bars are 1 standard deviation: REMD_hPTH1, REMD_hPTH2, REMD_hPTH3, REMD_hPTH4, and REMD_hPTH5.

8.3.2.1 Untrained SOMs by C_v Comparison

The C_v distributions for untrained SOMs could detect differences between the MET conformational datasets if the maxmin initialization was used (Section 7.3.1, Figure 16); this implies that if the hPTH/REMD data-subsets are different, the untrained C_v distributions will show a difference. Figure 45 shows the untrained SOMs (Table 13:8-a to 8-o) produced using the two different programs (C++ and MATLAB) and two different nodal geometries (rectangular and hexagonal) and maxmin initialization. The C_v distributions overlap for untrained SOMs of all five data-subsets regardless of the program used or nodal geometry. These data-subsets are considered similar.

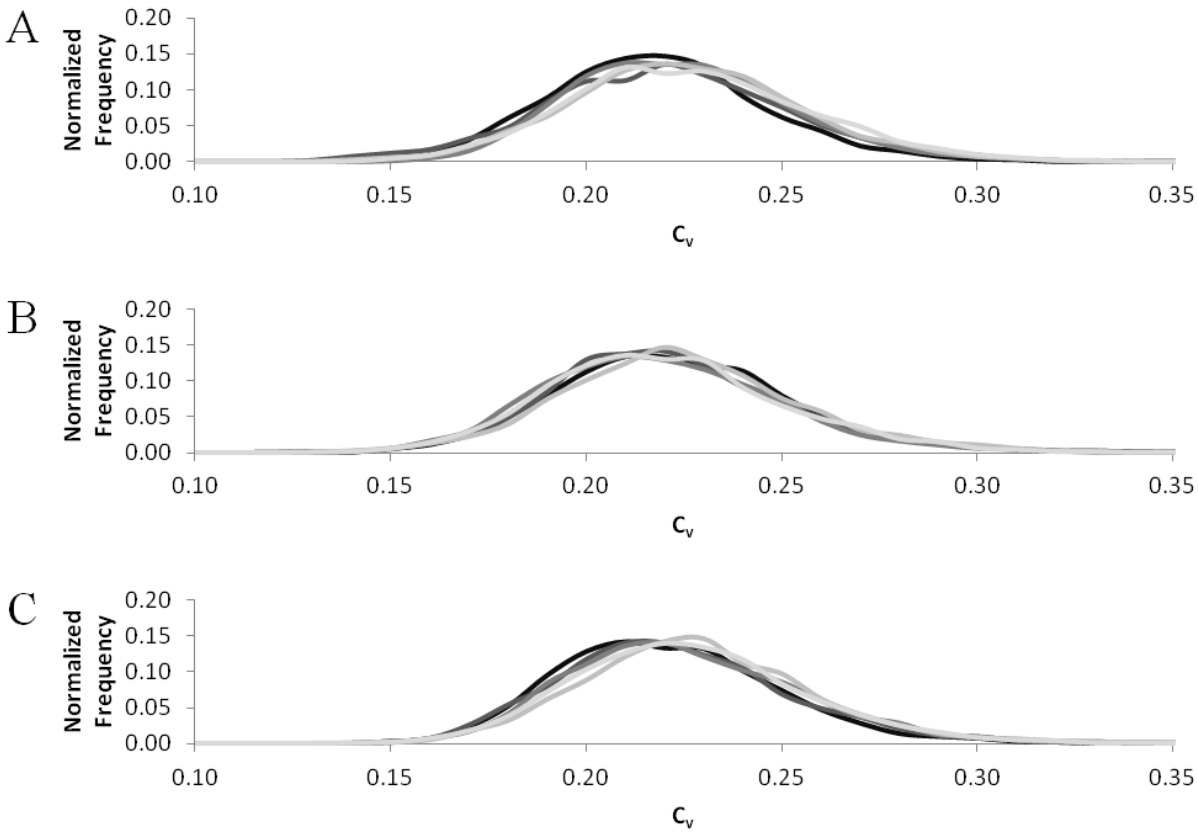


Figure 45: Untrained 6×6 SOMs generated from 108-dimensional data-subsets (—) REMD_hPTH1, (—) REMD_hPTH2, (—) REMD_hPTH3, (—) REMD_hPTH4, and (—) REMD_hPTH5. (A) 100 rectangular SOMs generated from the C++ program (Table 13:8-a to 8-e), (B) 100 rectangular SOMs generated from the MATLAB program (Table 13:8-f to 8-j), and (C) 100 hexagonal SOMs generated from the MATLAB program (Table 13:8-k to 8-o).

8.3.3 Optimizing Training Length of the SOM Programs

The parameters used for training the SOMs are based on the findings from the SOM analysis of MET (Section 7.3). The SOMs were run 100 times and have toroidal boundaries (equation [6]) and were trained by a sequential training algorithm (equation [12]) and a decreasing neighbourhood function (equation [9] and [10]). The MATLAB program uses the LINEAR learning rate algorithm (equation [16]) and a learning rate factor of 0.05. The training length of the SOM must be optimized for each program. Since the C++ and MATLAB programs are trained by different neighbourhood functions, they may require different training lengths to produce optimal clusters. The SOM programs C++ and MATLAB train the SOMs by a

sequential training algorithm (Equation [12]). The algorithm finds the BMU for a randomly selected conformation and then updates the BMU nodal centre and those for the nodes in the neighbourhood. Each re-assignment and update for the randomly selected conformation is an iteration. When the number of iterations is equal to the training length, the SOM is considered trained.

There is no default adjustment/dependency of the sequential training algorithm on the dataset dimensionality anticipating its impact on the training length. If the conformational dataset being trained on the SOM is for a protein that is 5 amino acids in length or 34 amino acids in length, no change in training length is made by the SOM algorithms. The proteins used in this work are flexible. Increasing the size of the protein increases the span of conformational space, which could require an increase in the training length to produce optimal clusters.

Figure 46 shows C_v distributions of trained 6×6 SOMs generated from the C++ program from the 108-dimensional REMD_hPTH1 data-subset (Table 13: 8-p to 8-u). The training lengths (100, 500, 1000, 5000, 10000, 15000) have little impact on the similarity of data partitioning produced. The training lengths 5000, 10000, and 15000 minutely shift the majority complement region to the right. The training length of 10000 is chosen for subsequent SOM training. There is a small peak located at ~0.90 for all of the different training lengths; this region represents SOMs which have similar partitioning.

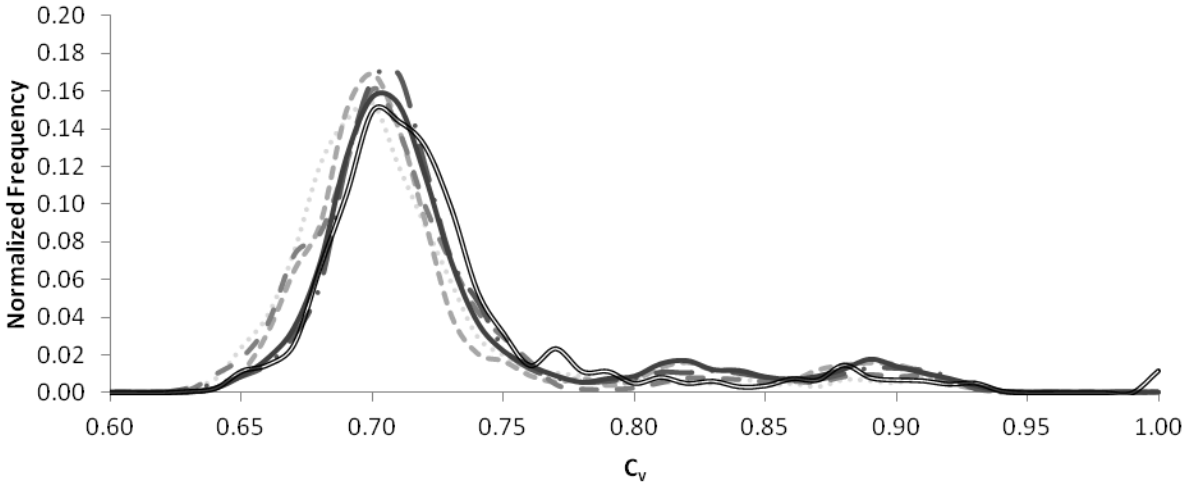


Figure 46: The C_v distributions of 6×6 SOMs trained by different training lengths generated from the 108-dimensional REMD_hPTH1 data-subset using the C++ program. The SOMs were trained with different training lengths: (.....) 100 (Table 13:8-p), (- - -) 500 (Table 13:8-q), (– –) 1000 (Table 13:8-r), (— .) 5000 (Table 13:8-s), (—) 10000 (Table 13:8-t), and (—) 15000(Table 13:8-u).

Figure 47 shows C_v distributions of trained 6×6 SOMs generated using the MATLAB program and the 108-dimensional REMD_hPTH1 data-subset (Table 13:8-v to 8-bb). The training lengths (100, 500, 1000, 5000, 10000, 15000, and 20000) have a greater impact on the C_v distributions generated from the MATLAB program than from the C++ program. To determine the training length that will be used, the peaks were compared to determine which training lengths produced similar C_v distributions. If the training length is optimized, no matter how many more training steps are added, the C_v distribution should not change. The C_v distributions produced by 15000 and 20000 have similar peaks. The lowest value was chosen as the optimal training length.

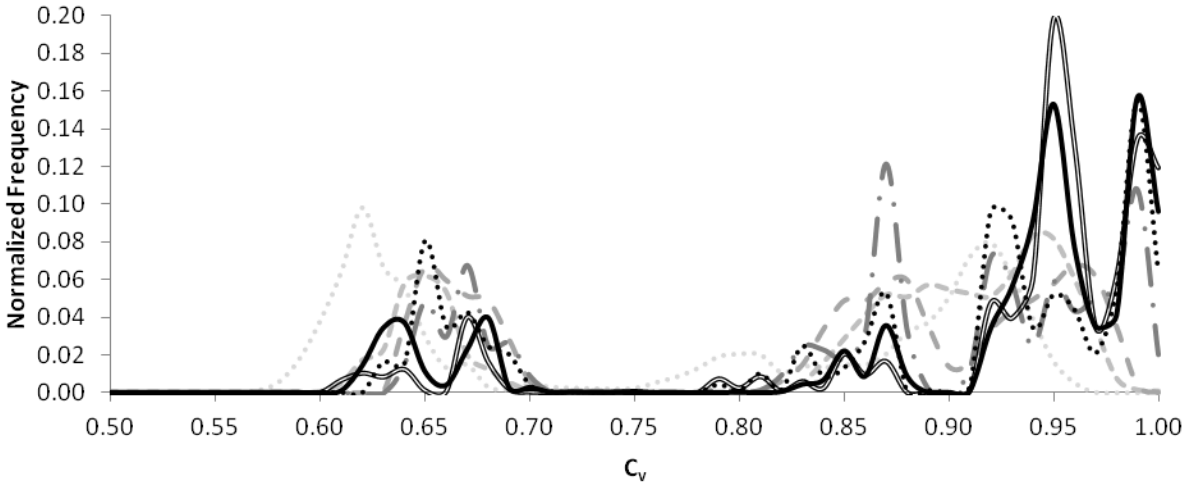


Figure 47: The C_v distribution of trained 6×6 SOMs generated from the 108-dimensional REMD_hPTH1 data-subset using the MATLAB program. The SOMs were trained with different training lengths: (.....) 100 (Table 13:8-v), (- - -) 500 (Table 13:8-w), (– –) 1000 (Table 13:8-x), (——) 5000 (Table 13:8-y), (.....) 10000 (Table 13:8-z), (——) 15000 (Table 13:8-aa), and (——) 20000 (Table 13:8-bb).

A training length of 10000 was found to be optimal for 6×6 SOMs trained by the C++ program using the 108-dimensional data-subsets of 5000 conformations of hPTH. Conversely, the optimal training length was found to be 15000 for the MATLAB program, given the same datasets and mapsize. When the mapsize was increased to a 10×10 SOM trained by the C++ program, the optimal training length decreased to 5000 (Figure 48A). However, no high C_v values were found. These SOMs will not be discussed and further work will have to be done to determine if the 10×10 SOM trained by the C++ program produced similar clusters to the other SOMs. Conversely, the optimal MATLAB program training length was found to be 15000 for the 10×10 SOM (Figure 48B). There was no decrease in the training length; however, a high C_v region was found, this region contains two peaks. These training lengths were determined to generate SOMs with close to optimal reproducibility given 6×6 or 10×10 maps. Clustering the data using SOMs with different mapsizes would require a reassessment of training lengths.

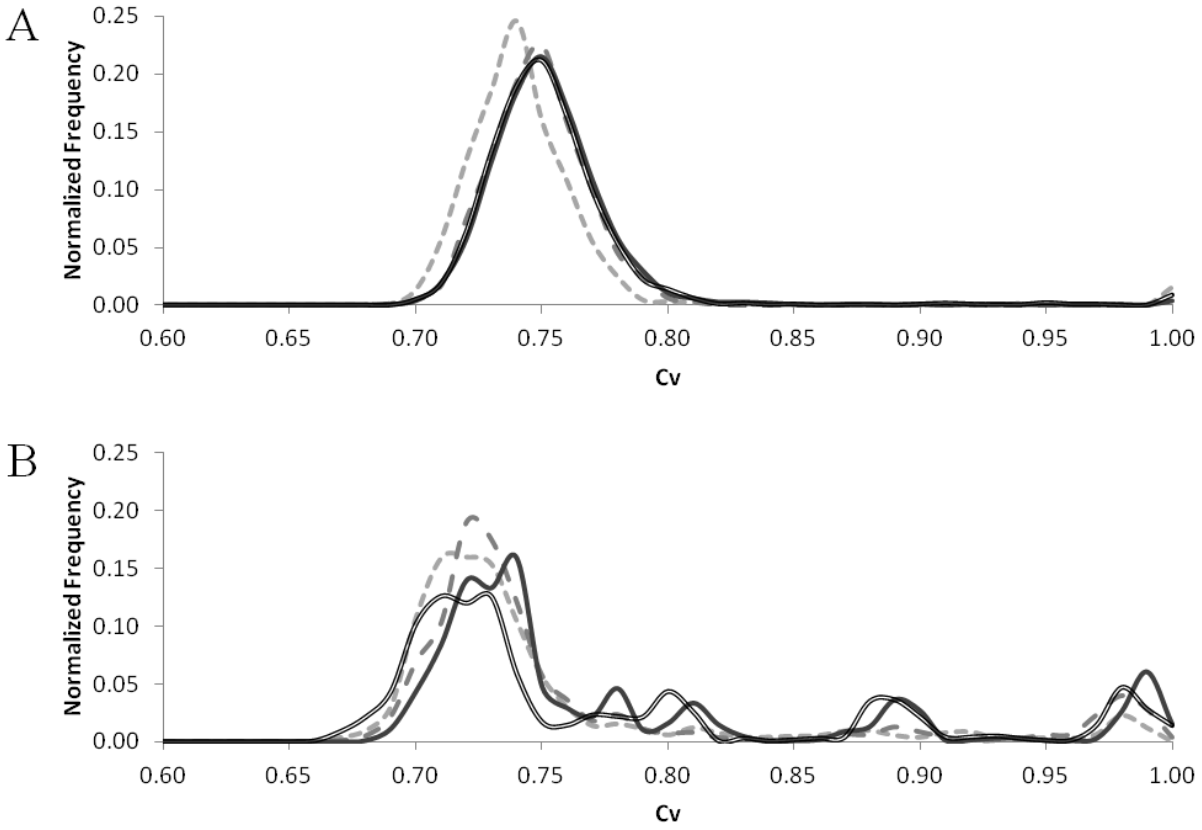


Figure 48: The C_v distributions of different training lengths of the 10×10 SOMs for the 108-dimensional REMD_hPTH4 data-subset. (A) Generated from the C++ program and trained with different training lengths: (- - -) 500 (Table 13:8-cc), (– –) 1000 (Table 13:8-dd), (——) 5000 (Table 13:8-ee), and (═══) 10000 (Table 13:8-ff). (B) Generated from the MATLAB program and trained with different training lengths: (- - -) 5000 (Table 13:8-gg), (– –) 10000 (Table 13:8-hh), (——) 15000 (Table 13:8-ii), and (═══) 20000 (Table 13:8-jj).

8.3.4 SOMs Trained by the C++ Program with the REMD Data-Subsets

The C_v distributions were computed for 100 rectangular 6×6 SOMs produced by the C++ program generated from each data-subset (Table 13:8-t, 8-kk to 8-nn) of the hPTH/REMD simulation (Figure 49). These distributions all have a peak around 0.7 that is in the majority complement region and around 0.9 that is in the high C_v region. Many of the SOMs have a separation between these peaks. The C_v distributions for SOMs produced from the subsets REMD_hPTH1 (Table 13:8-t) and REMD_hPTH3 (Table 13:8-ll) have a third peak between the majority complement region and the high C_v region. The trained SOMs for the data-subsets

REMD_hPTH4 (Table 13:8-mm) and REMD_hPTH5 (Table 13:8-nn) have well-defined separations between the majority complement region and the high C_v region in the C_v distributions.

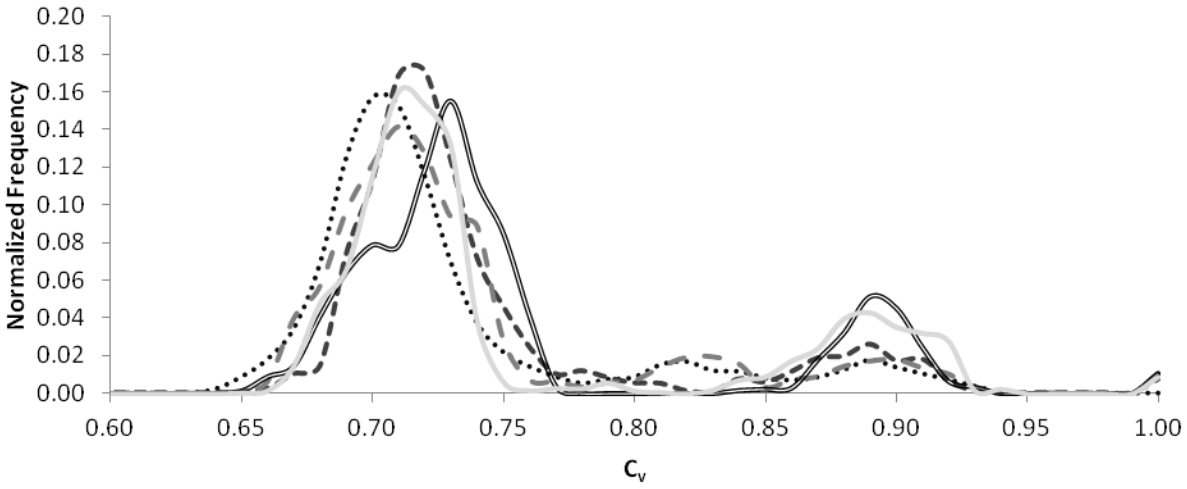


Figure 49: The C_v distributions of 100 rectangular 6×6 SOMs generated by the C++ program for the five data-subsets of hPTH: (.....) REMD_hPTH1 (Table 13:8-t), (– – –) REMD_hPTH2 (Table 13:8-kk), (– –) REMD_hPTH3 (Table 13:8-ll), (═══) REMD_hPTH4 (Table 13:8-mm), and (——) REMD_hPTH5 (Table 13:8-nn). Training length was 10000 iterations.

The SOMs which were trained by the data-subsets were used as a supervised classifier for the 541100 hPTH/REMD conformational dataset (Section 2). Figure 50 shows the average percent of the original 54110 conformations of hPTH/REMD classified by each of the SOMs generated from the five data-subsets. The average was computed over 100 SOMs and the error bars are for one standard deviation. Figure 50 shows that there is no difference between the ability of the SOMs produced from REMD data-subsets to classify the members of the larger 541100 hPTH/REMD dataset. It also shows that over 99% of the original conformations are classifiable by the trained SOMs

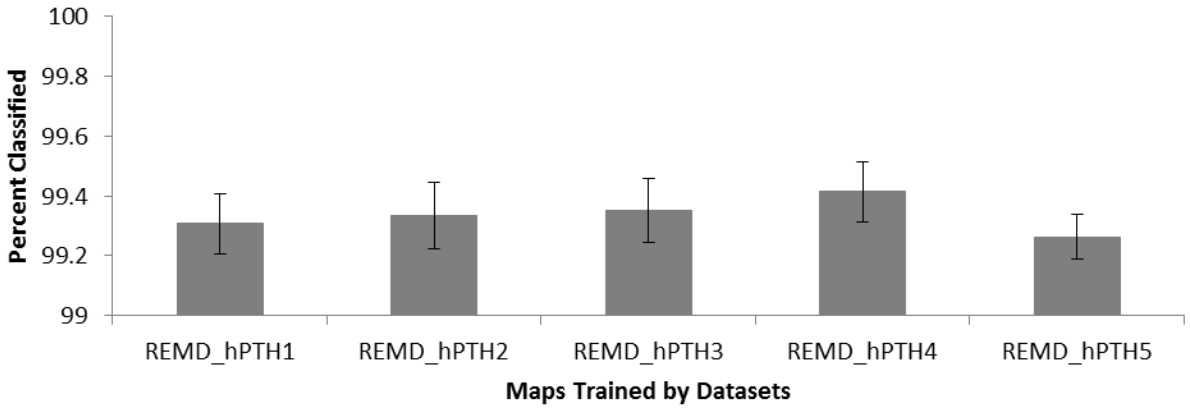


Figure 50: The percent of the original 541100 conformations of hPTH/REMD classified by the 6×6 SOMs trained from the 5 different data-subsets. There are 100 rectangular SOMs generated from the C++ program and trained for 10000 iterations for each of the 5 data-subsets (Table 13:8-t, 8-kk to 8-nn).

Therefore, SOMs produced from any of the data-subsets can be used to explain the conformations saved from the hPTH/REMD simulation (Figure 50). Due to the appearance of the C_v distribution (Figure 49), the data-subset REMD_hPTH4 appears to be the best choice. The data-subset REMD_hPTH4 and REMD_hPTH5 have the most maps in the high C_v region (>0.85). The data-subset REMD_hPTH4 was chosen for its shift in higher C_v values in the majority complement region.

8.3.4.1 Supervised Training SOMs for NMR Models

The ten independent classical MD simulations of hPTH were started from their corresponding NMR model conformations (1ZWA).[94] Chu *et al.* [97] and Gordon and Somorjai[3] found the classical MD simulations deviated from the original starting conformation and classical MD simulations did not cover all conformational space. The NMR models are not very similar to the classical MD simulations, including the ones from which they started from.

The REMD simulation was started from the NMR model 2 of 1ZWA. When using trained SOMs produced from the five different data-subsets of the hPTH/REMD simulation as supervised classifiers, many of the ten NMR models were not readily accepted as members of the

SOMs' nodes. Figure 51 shows the percentage of the SOMs which could accept the NMR models as nodal members. Models 1, 3, 4, 5, and 10 were rarely classified. Model 2 was the NMR conformation that was most similar to those members in clusters found in hPTH/REMD simulation. Models 6, 7, 8 and 9 were also classified by many of the SOMs. The NMR models were refined using their 206 NOESY cross peaks as distance restraints and a MD simulation was done using simulated annealing.[94] The refinement dielectric constant was set to 4.[94] This dielectric constant is not consistent with an aqueous environment but rather was chosen to reflect the dielectric constant throughout the hydrophobic areas of the protein.[98] The use of this value for the dielectric constant could have generated unrealistic structures for an aqueous environment. Many of these original NMR models were rejected from the nodal membership of the SOMs trained by the conformations produced from the MD and REMD simulations of hPTH in explicit aqueous solvent.

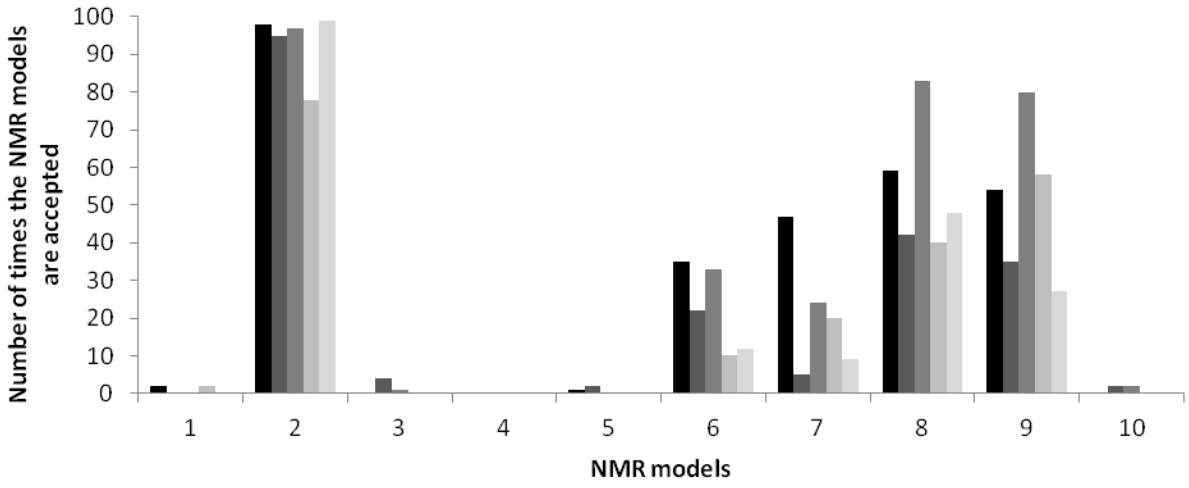


Figure 51: The hPTH/REMD number of times each of the 10 NMR models of 1ZWA can be classified by 100 SOMs trained using the 5 different data-subsets. The rectangular 6×6 SOMs were trained for 10000 iterations using the C++ program. ■ REMD_hPTH1 (Table 13:8-t), ■ REMD_hPTH2 (Table 13:8-kk), ■ REMD_hPTH3 (Table 13:8-ll), ■ REMD_hPTH4 (Table 13:8-mm), and ■ REMD_hPTH5 (Table 13:8-nn).

8.3.4.2 Comparing X-ray Structures to hPTH conformations obtained from REMD Simulations

The PDB file 3C4M is an X-ray crystallographic structure of the N-terminal extracellular domain of the PTHR with the C-terminal portion of the hPTH (15-34).[92] This X-ray structure contains only the extracellular portion of the PTHR bound to the corresponding portion of the hPTH (Figure 41). The X-ray structure is a dimer and both hPTH conformations are used in the following analysis. The hPTH fragments were compared to each other by RMSD of the backbone atoms, which is 0.418Å. The trained SOMs described in Section 8.3.4 were used as a supervised classifier for these two X-ray structures. The goal was to ascertain if the bound 15-34 C-terminal portion of hPTH was a conformational subset of the free conformations of the 1-34 hPTH explored by the REMD simulations.

Figure 52 shows a 6×6 rectangular SOM, one of the X-ray structures of the PTHR-bound hPTH, and the sequence of hPTH. The sequence contains 34 amino acids and the N-terminus is tagged blue and the C-terminus is tagged red. The dark grey shows the amino acids used in the training of the SOMs and the orange circles show the amino acids that are found in the X-ray structure. The SOM shows the occupancy (number of conformational members) of the nodes by the relative size of the squares and the conformation closest to the nodal centre.

A program was written to determine if the X-ray structures would be accepted into the nodal membership of each node of the trained SOMs. Unlike the previous ways that supervised clustering with an SOM was done, this supervised clustering could accept each conformation into all nodes on the SOM. Only the components of the dihedral angles from amino acids 15-31 were used. A new neuron tolerance was computed to determine the conformation of the nodal membership that was the farthest away from the nodal centre, when only considering amino

acids 15-31. If the X-ray structure was closer to the nodal centre than the neuron tolerance, the X-ray structure was accepted into the nodal membership. The SOM in Figure 52 shows the 26 nodes that have accepted the X-ray structures into the nodal membership; these nodes have the amino acids 15-31 coloured orange. This suggests that a significant proportion (0.80) of the unbound hPTH consists of conformations suitable for binding to the extracellular PTHR. That is, the first phase of binding can be described as "conformational selection", where a receptor binds to its flexible ligand, which has pre-folded into a suitable binding shape.[86] Nothing can be ascertained about the second binding phase of the N-terminus of hPTH to the juxtamembrane PTHR region since there is no experimental structural data with which to compare our simulated hPTH conformations.

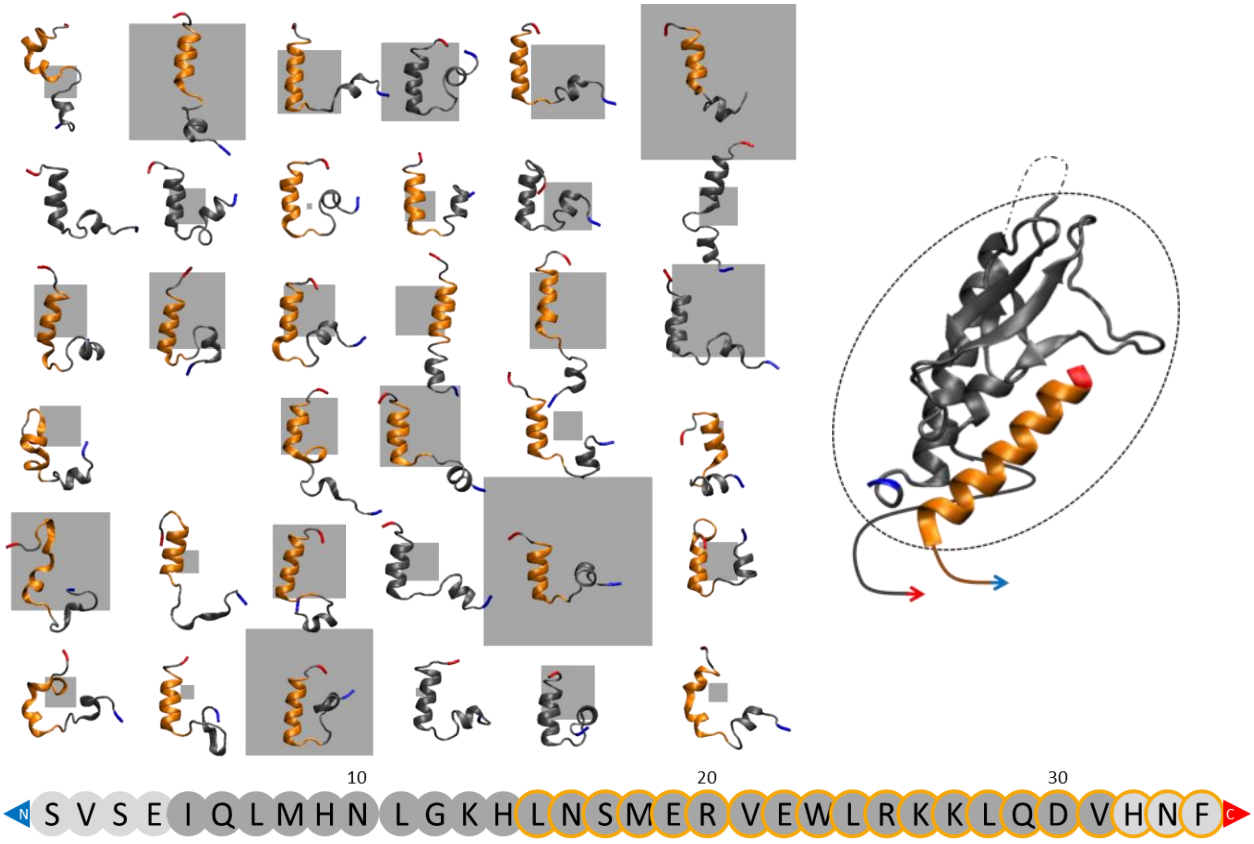


Figure 52: The sequence of hPTH (bottom) and an example of an SOM trained by hPTH dataset. The sequence contains all 34 amino acids and the N-terminus is tagged blue and the C-terminus is tagged red. The dark grey shows the amino acids used in the training of the SOMs. The orange circles show the amino acids present in the X-ray structure (upper right). The rectangular 6×6 toroidal SOM was trained for 10000 iterations from the REMD_hPTH4 data-subset from the C++ program (upper left) (Table 13:8-mm). The size of the squares is representative of the occupancy of the node. The images of hPTH correspond to the conformations closest to the nodal centres. The C and N-terminus are tagged similarly to the sequence. The orange sections in these images are of amino acids (15-31), which is the overlap of the X-ray structure and the amino acids used in training. The orange section is only shown for nodes which accept one of the X-ray structures into their nodal memberships.

8.3.4.3 Comparing Classical MD Ensembles to the REMD Ensembles of hPTH by SOMs

The classical MD simulations are not as broad in terms of conformational diversity as the REMD simulations. Classical MD simulations can get stuck in potential energy wells that are separated by large barriers, whereas REMD simulations use parallel simulations run at different temperatures among which occasional exchanges of conformations allow the low temperature simulations to sample conformational space accessible to the higher temperature simulations. One of the ways to improve the sampling of classical MD simulations is by using starting

conformations from various poses from solved NMR solution structures. All ten models from 1ZWA were used as starting conformations of classical MD simulations that were run for 2 ns (Section 8.2.1). The 5125 conformations saved from each of these runs were then classified using the 100 trained 6x6 rectangular SOMs generated from the C++ program and the REMD_hPTH4 data-subset (Figure 53). If the MD simulations do not explore as broad a range of conformational space as the REMD simulations, we expect that they will be explained by a subset of nodes on the SOM trained on the REMD simulation (Figure 54).

Figure 53 shows the percent of conformations saved from the classical MD simulations that could be classified by the SOM trained on the REMD_hPTH4 data-subset (Table 13:8-mm). Many of the classical MD simulation conformations (2, 5, 6, 7, 8, 9, and 10) are described by the REMD_hPTH4 data-subset. However, hPTH/MD1, hPTH/MD3, and hPTH/MD4 are not well described. This is either because the REMD simulation did not explore all conformational space or the starting conformations of these simulations (models 1, 3, and 4) of 1ZWA are not realistic representations of hPTH in an aqueous environment and these simulations remained stuck in unrealistic phase space.

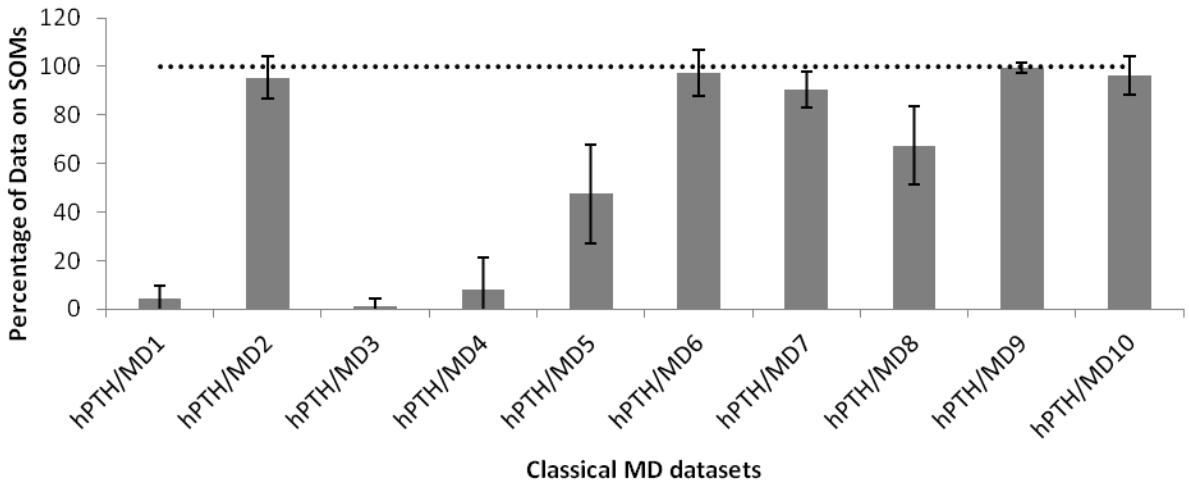
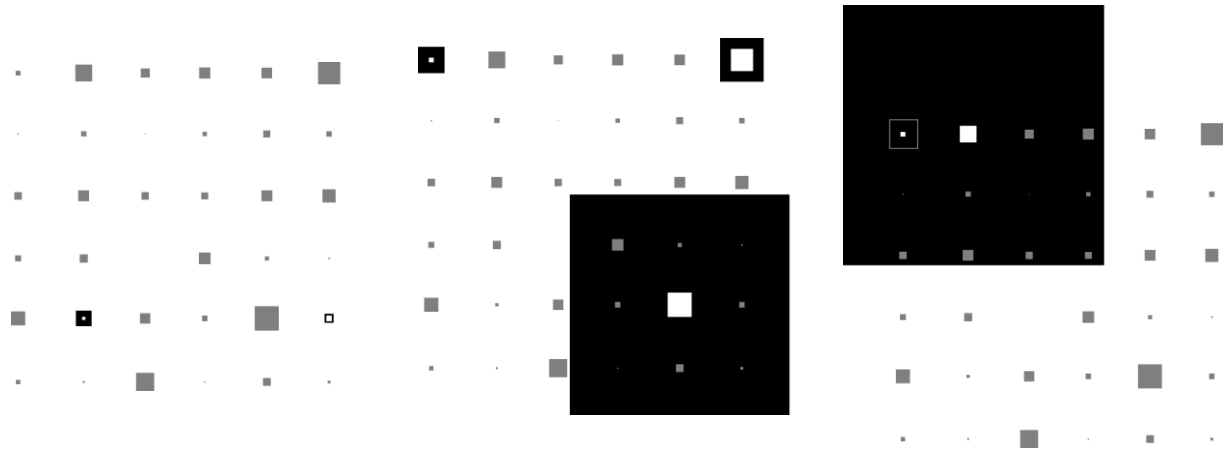


Figure 53: The percentage of each of the 108-dimensional datasets of 10 MD conformational datasets plotted on SOMs generated from REMD_hPTH4 data-subsets. There are 100 rectangular SOMs generated from the C++ program and trained for 10000 iterations (Table 13:8-mm). The dotted line is 100%.

Figure 54 shows three examples of classical MD simulations classified by a 6×6 rectangular SOM that was generated from the C++ program and trained for 10000 iterations. The size of the nodes shows the occupancy within the nodes. The white and grey nodes represent the original REMD SOM. A white node shows when the REMD SOM can classify the hPTH/MD simulations. The black nodes show the proportion of the classical MD simulation on the REMD SOM. As expected, the classical MD simulations MD2 and MD10 are explained by a subset of nodes on the SOM trained using the REMD simulation conformational data. That is, a very large proportion of MD2 and MD10 conformations are described by a subset of one to three REMD conformational clusters.



hPTH/MD1 hPTH/MD2 hPTH/MD10

Figure 54: Three different MD datasets plotted on a 6×6 rectangular SOM generated from REMD_hPTH4 data-subsets. The 6×6 rectangular SOM was generated from the C++ program and trained for 10000 iterations (Table 13:8-mm). The grey nodes are the number of conformations in the original REMD SOM. The white nodes are the REMD SOM for a node that could classify the conformations from the classical MD simulation. The black node shows the proportion of MD simulation conformations which can be classified by the REMD SOM.

Using Figure 51, Figure 53, and Figure 54, a comparison will be made to determine if the NMR structures might not be realistic representations of hPTH in an aqueous environment and these simulations remained stuck in unrealistic phase space. Figure 51 and Figure 53 shows that the NMR structure 2 and the classical MD simulation hPTH/MD2 can be classified by the REMD SOM. Furthermore, Figure 54 shows that the hPTH/MD2 simulation is classified by only a few nodes of the SOM produced from the REMD/hPTH4 data-subset. Figure 51 shows that the NMR structures 1 and 10 cannot be plotted onto the REMD SOM very frequently. However, when an MD simulation is done starting from model 10, the conformations produced from that simulation can be classified by the REMD simulation, whereas conformations from the MD simulation commenced with model 1 are still not well classified by the REMD SOM. This means model 1 in the NMR is probably not a realistic representation of hPTH in an aqueous environment and the MD simulation (hPTH/MD1) could not get out of the unrealistic space. However, model 10 in the NMR is also probably not realistic representations of hPTH in an

aqueous environment but the MD simulation (hPTH/MD10) did move into a region explored by the REMD simulation.

8.3.4.4 Reproducible Clusters of hPTH in 6×6 rectangular SOMs

As seen in Chapter 7, Met-Enkephalin (MET) (Section 7.3.4 and 7.3.9), the data partitioning produced by SOMs can be reproducible when run multiple times. However, for the conformational datasets, "multiple" means $\Theta(10^2)$. To assess reproducibility, an objective function is used to give a numerical quantity to the quality of partitioning of the dataset onto the two-dimensional map, as well as C_v to evaluate the similarity of the partitioning. Figure 55 presents a scatter graph of the objective functions (NEIGH or INSSQ) versus their C_v values. There are 100 toroidal 6×6 rectangular SOMs compared to each other (100×100 comparisons). These particular SOMs produce two predominant partitionings. As long as the training length is adjusted for the protein (size/flexibility), reproducible clusters can still be found, as seen by groups 1 and 2 in Figure 55.

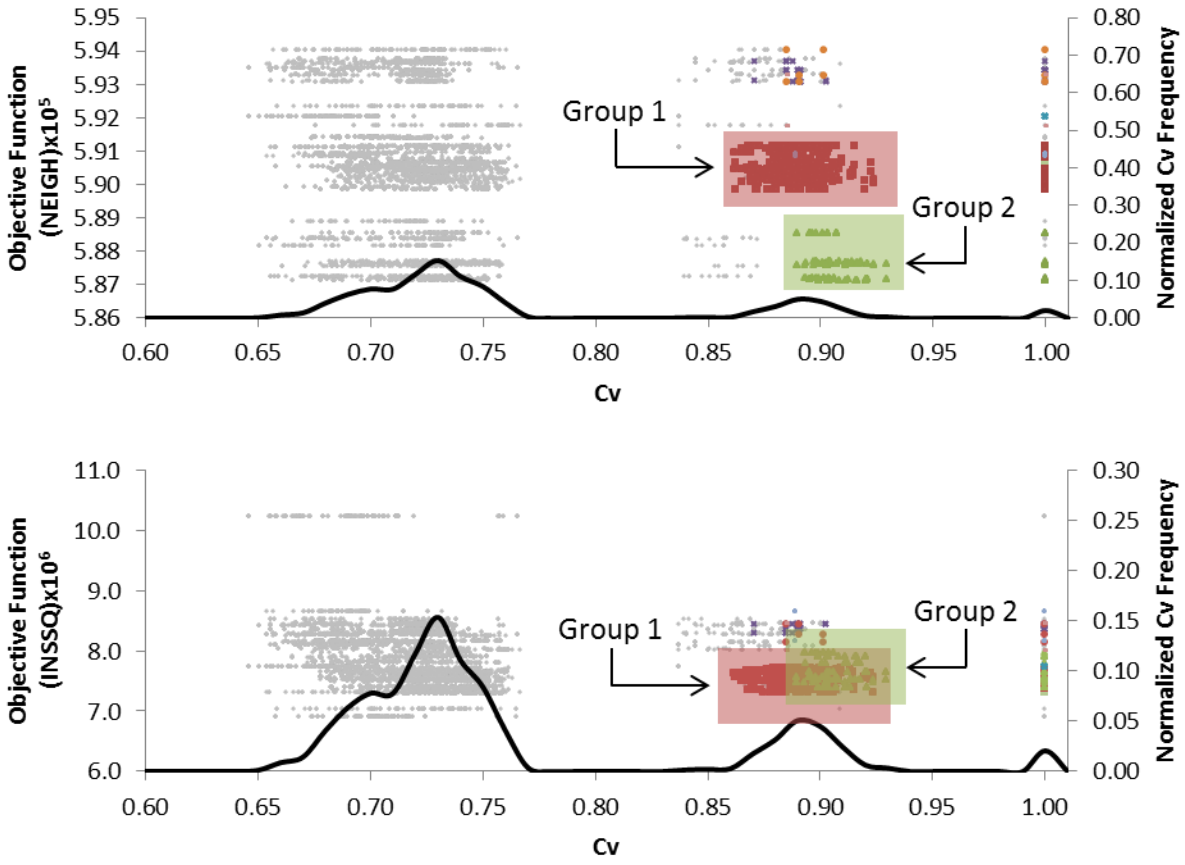


Figure 55: The comparison of objective functions to C_v values generated from the 100 toroidal 6×6 rectangular SOMs from the C++ program with the REMD_hPTH4 data-subset (Table 13:8-mm). The scatter graphs are of the NEIGH objective function (top) and the INSSQ objective function (bottom). The C_v distribution (—) is along the bottom of the graph with the normalized values on the right vertical axis. The C_v is generated from two maps; for each map an objective function was computed (one C_v value has two objective functions). ◆ are all the C_v values, the coloured points are the 8 groups of maps which produce high C_v values with each other and have 3 or more maps in a group. ■ is the highest occupied group with 38 SOMs(group 1), and ▲ is the second highest occupied group with 14 SOMs (group 2).

This method found two reproducible and different partitionings of the hPTH/REMD4 conformational dataset; they are labelled by the number of SOMs in the group. Group 1 contains 38 SOMs while Group 2 contains 14 SOMs (Figure 55). These same two groups of data partitionings were found for two different objective functions. Our program separates groups of like SOMs by high C_v values; if their objective functions are close in numerical value it is found out after the groups are generated. The mean C_v value is larger for Group 2 than Group 1. The SOMs in Group 2 have lower values of the NEIGH objective function than those of Group 1.

The SOMs in Group 1 and Group 2 have approximately equal values of the INSSQ objective function. The probably means Group 1 and Group 2 have a different neighbourhoods while their nodal memberships are similar. This method was able to find two reproducible partitions of the data.

8.3.4.4.1 Partitioning of Clusters and Experimental Data

From Figure 55 it was determined that there are two different partitionings of the 5000 member of hPTH/REMD4 conformational data-subset. The remaining question is whether they represent completely different partitioning of the data or one is a subset of the other. One SOM was taken from each group. When the SOMs were compared to each other, the result was a C_v value of 0.74, which is in the majority complement region of Figure 55. When the maps are compared by their conformations closest to the nodal centre, six of the 56 nodes have the same conformations.

The individual values of ${\chi^2}_{ij}$ compare nodes from one partitioning to another (Section 2.5.1). If the partitions are very similar, there should be a one-to-one correspondence between them, that is, there would be 36 values of high ${\chi^2}_{ij}$ and (36^2-36) near-zero values of ${\chi^2}_{ij}$. If the partitionings are not similar, the χ^2 test is able to show the relationship between the SOMs. Figure 56 shows 200 ${\chi^2}_{ij}$ values (1296 comparisons in total) for the two 6×6 SOMs, one from Group 1 and the other from Group 2. The ${\chi^2}_{ij}$ values are organized by decreasing value. The largest 23 ${\chi^2}_{ij}$ values are highlighted. These were chosen because of the steeper decline after the 23^{th} ${\chi^2}_{ij}$ value.

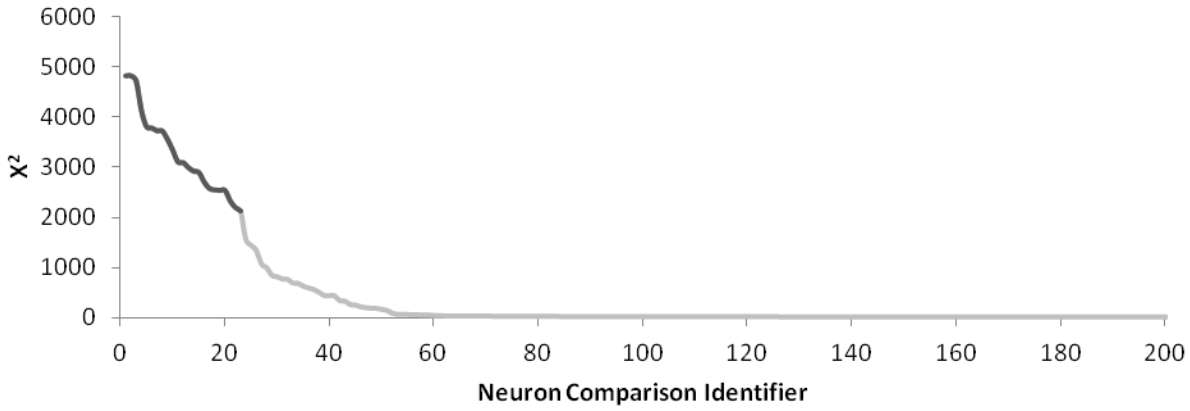


Figure 56: The χ^2_{ij} comparison between all the nodes of a 6×6 SOM of Group 1 to all nodes of a 6×6 SOM of Group 2. The Neuron Comparison Identifier is an assigned label. The top 23 values are highlighted in dark grey.

The χ^2_{ij} values were used to correlate the nodes between the different SOMs and to determine if one of the partitionings is a subset of the other. If the Group 1 SOMs were a subset of the Group 2 SOMs the more diverse clusters from Group 2 SOM could not be divided into smaller groupings. If the data partitioning of the Group 1 and Group 2 SOMs are completely different, the SOMs are distributing the data differently across many nodes. The top 23 χ^2_{ij} values were used to correlate the nodes between the two SOMs; analogous nodes can be seen by the different colours shown in Figure 57. Figure 57 shows the two SOMs, the nodes are coloured based the similarity of the nodes based on χ^2_{ij}values; the same colour of the node shows a similar nodal membership in both SOMs. Group 1 has a similar partitioning to Group 2. All clusters in Group 1 which share memberships (two colours) are neighbours on Group 2.

Group 1 Group 2

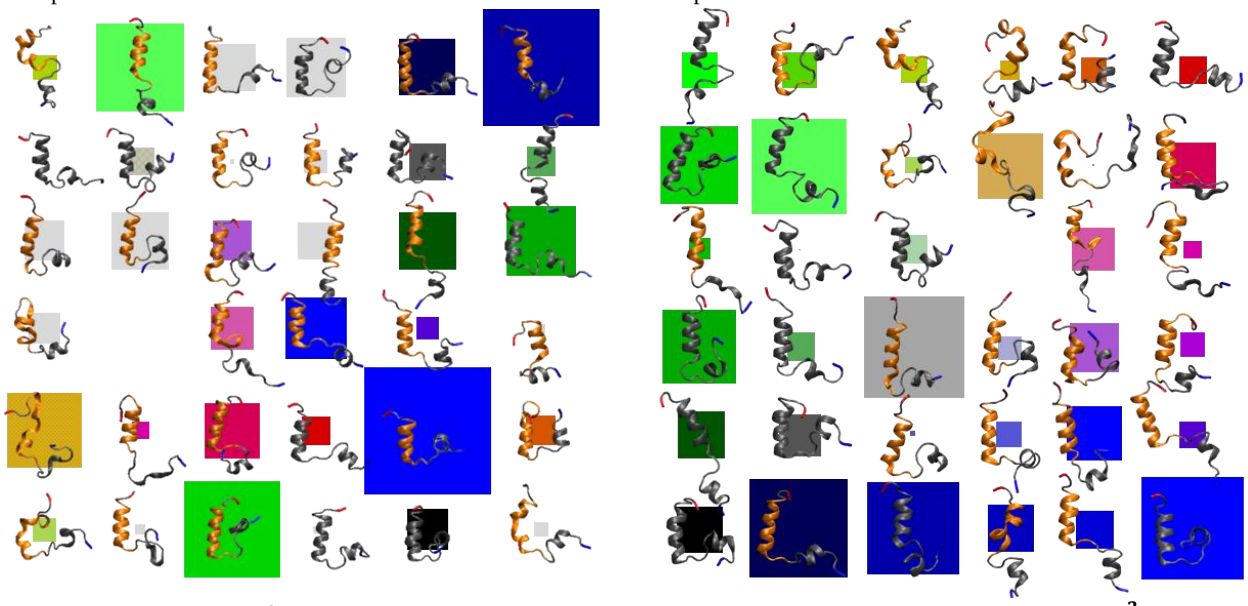


Figure 57: The colours of the nodes are assigned in the Group 2 SOM and correlate to Group 1 by 23 χ^2 values. The SOMs are toroidal 6×6 rectangular SOMs generated from the REMD_hPTH4 data-subset (Table 13:8-mm) using the C++ program. These images are similar to Figure 52. The size of the square is representative of the occupancy of the nodes. The image corresponds to the closest conformation to the nodal centre. The orange sections in these images are of amino acids (15-31), which is the overlap of the X-ray structure and the amino acids used in training. The orange section is only shown for nodes which accept one of the X-ray structures into its nodal membership.

8.3.4.5 Reproducible Clusters of hPTH in 10×10 Rectangular SOMs

In Section 8.3.4.4 it was found that the 6×6 SOM trained by the C++ program generated two different and reproducible partitionings of the hPTH/REMD data-subset. However, when the mapsize is increased there are no reproducible clusters. Figure 58 shows the C_v distributions for the trained and untrained 6×6 and 10×10 rectangular SOMs produced by the C++ program. As expected, the C_v distributions for untrained 10×10 SOMs are shifted to the right of the untrained 6×6 SOMs. This shift is due to the increase of nodes among which the conformations can be distributed. The C_v distribution for trained SOMs with exhibit a majority complement region as well as a high C_v region for the 6×6 SOMs, while that for the 10×10 SOMs only have the majority complement region. There are no reproducible data partitionings of the hPTH/REMD4 data-subset using the C++ program to produce 10×10 SOMs.

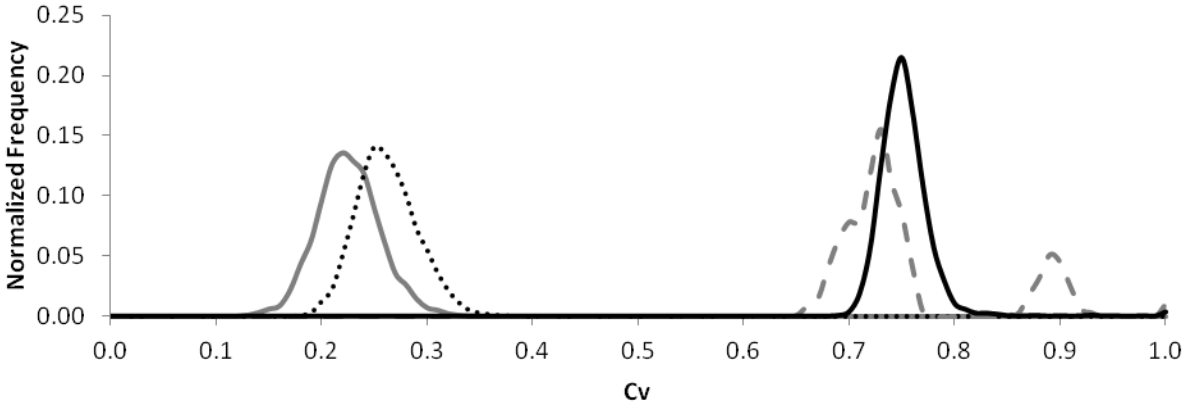


Figure 58: The C_v distributions for the untrained and trained 6×6 and 10×10 SOMs generated by the C++ program for the 108-dimensional REMD_hPTH4 data-subset. (——) 6×6 untrained SOM (Table 13:8-d), (.....) 10×10 untrained SOM (Table 13:8-oo), (– – –) 6×6 SOM trained for 10000 steps (Table 13:8-mm), and (——) 10×10 SOM trained for 5000 steps (Table 13:8-ee).

8.3.5 SOMs Trained by the MATLAB Program with the hPTH REMD4 Data-Subset

The MATLAB program was used as a secondary source code to compare trained SOMs produced by independent computer programs. The same hPTH/REMD4 data-subset was used to train the MATLAB SOMs. Unlike the C++ program, the MATLAB program can use different nodal geometries (rectangular and hexagonal). The results from analysis of SOMs produced for conformations of MET (Section 7.3.9) show that reproducible SOMs can be found for a small peptide independent of the source code. Furthermore, the ability of the C++ Program to find reproducible clusters (Section 8.3.4.4) is not necessarily dependent on the size (number of amino acids) of the protein. In this Section, the size of the protein or the nodal geometry influences the ability of the MATLAB program to find reproducible SOMs.

Figure 59 shows examples of both hexagonal and rectangular 6×6 toroidal SOMs trained for 15000 iterations from the REMD_hPTH4 conformational data-subset using the MATLAB program (Table 13:8-pp and 8-qq, respectively). The size of the hexagons and squares demonstrate the occupancy (number of members in a cluster) of the nodes and the conformation closest to each nodal centre is illustrated. If one of the X-ray structures was accepted into the

nodal membership, the nodal centre images have amino acids 15-31 shown in orange. The comparison of the 100 hexagonal and rectangular 6×6 SOMs is discussed in Section 8.3.5.1

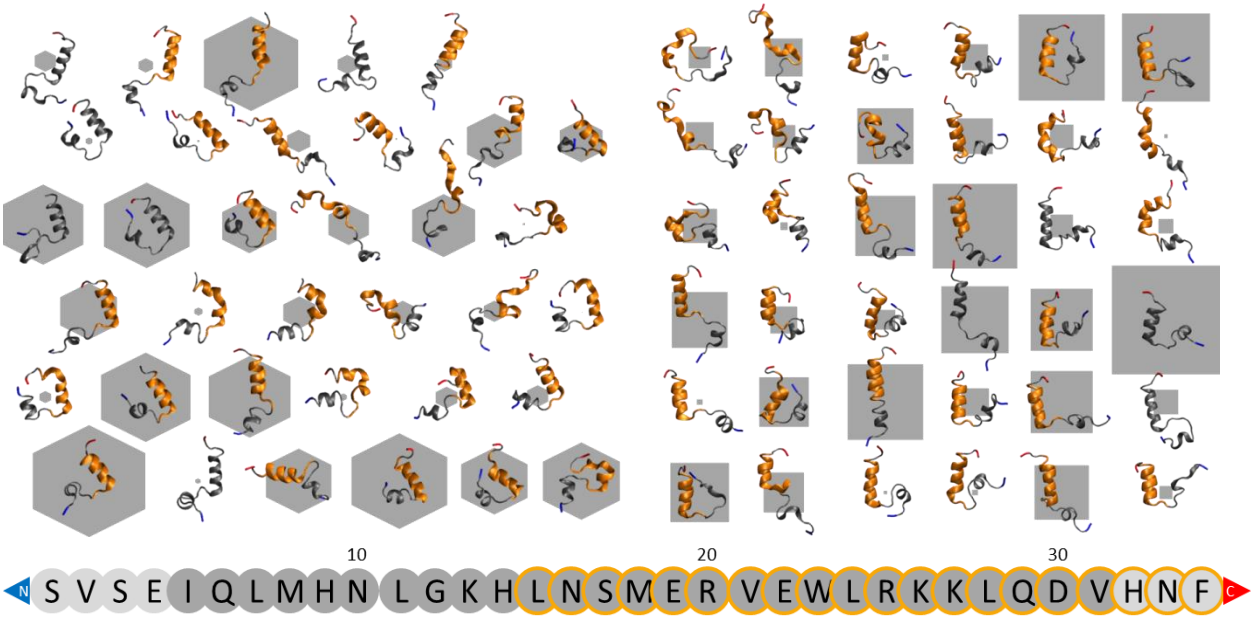


Figure 59: Examples of hexagonal and rectangular SOMs trained by MATLAB. Sequence (bottom) contains 34 amino acids and the N-terminus (blue) and the C-terminus (red). The dark grey shows the amino acids used in the training of the SOMs. The orange circles show the amino acids present in the X-ray structure. The 6×6 toroidal SOMs (hexagonal (right, Table 13:8-pp) or rectangular (left, Table 13:8-qq)) were trained for 15000 iterations from the REMD_hPTH4 data-subset from the MATLAB program. The size of the hexagon or square is representative of the occupancy of the node. The image on the node corresponds to the closest conformation to the nodal centre. The orange section is only shown for nodes which accept one of the X-ray structures into its nodal membership.

8.3.5.1 Reproducible Clusters of hPTH in 6×6 Rectangular and Hexagonal SOMs

Reproducible 6×6 but not 10×10 SOMs were found for the 108-dimensional hPTH/REMD4 conformational data-subset using the C++ Program (Section 8.3.4.4). To determine if this finding is dependent on the program used for training the SOM, the MATLAB program was used to do the same evaluation. The objective function (INSSQ) was used to give a quantitative measure of the quality of the partitioning of the dataset onto a two-dimensional map and C_v was used to evaluate the similarity between independently obtained maps.

8.3.5.1.1 6×6 Hexagonal SOMs Generated by the MATLAB Program

The 6×6 hexagonal SOMs generated using the MATLAB program for the REMD_hPTH4 data-subset (Table 13:8-pp) were partitioned into two groups by the C_v comparison. Figure 60 shows the two groups as well as all the C_v for the 100 SOMs. The C_v distribution shows a large gap between the majority complement region and the high C_v values. The high C_v value peak is located above 0.95 and the majority complement region is between 0.60 and 0.75. One of the groups contains 72% of the SOMs, while the other only contains 3% of the SOMs. When SOMs from each of these groups are compared, the C_v values are in the majority complement area. Since the second group contains only three SOMs, only the first group is considered reproducible. Therefore, the 6×6 hexagonal SOMs generated using the MATLAB program using the REMD_hPTH4 data-subset, produced the same data partitioning for 72 of the 100 independent runs.

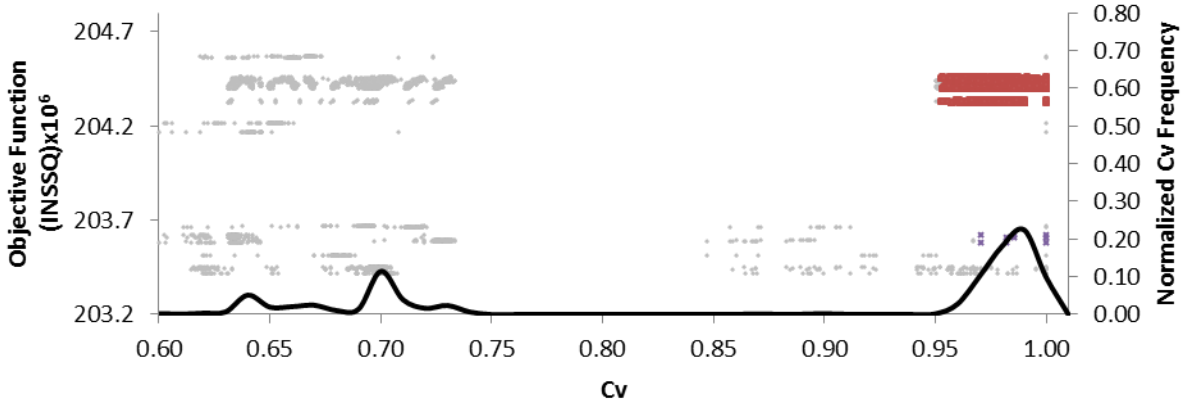


Figure 60: The INSSQ objective function versus C_v values for the 100 toroidal 6×6 hexagonal SOMs trained by the MATLAB program with the REMD_hPTH4 data-subset (Table 13:8-pp). The scatter graph is of the INSSQ objective function. The C_v distribution (—) is along the bottom of the graph with the normalized values on the right vertical axis. The C_v is generated from two maps; for each map an objective function was computed (one C_v value has two objective functions). ◆ are all the C_v values, the coloured points are the two groups of maps which produce high C_v values with each other and have three or more maps in a group. ■ is the highest occupied group with 72 SOMs (group 1), and ✖ is the second highest occupied group with 3 SOMs (group 2).

8.3.5.1.2 6×6 Rectangular SOMs Generated by the MATLAB Program

The 6×6 rectangular SOMs generated using the MATLAB program using the REMD_hPTH4 data-subset (Table 13:8-qq) was partitioned into six groups by the C_v comparison. Figure 61 shows the six groups as well as all the C_v values for the 100 SOMs. The C_v distribution shows a large gap between the majority complement region and the high C_v values. The majority complement region contains very few SOM comparisons and is located ~0.71. The high C_v values have a broad distribution between 0.82 and 1.00. There are approximately three peaks in this range of C_v. The six groups were found. Two of these groups contain 3% of the SOMs and are not considered reproducible. The other four groups are considered reproducible. However, when the C_v values of these four groups are compared to each other all the comparisons produced C_v values in the high C_v value range (>0.85). The 92 SOMs are considered to represent minor variations of the same data partitioning and therefore might merged if the training length was extended. The training length was only optimized for the hexagonal SOM, which has more nodes in the nodal neighbourhoods.

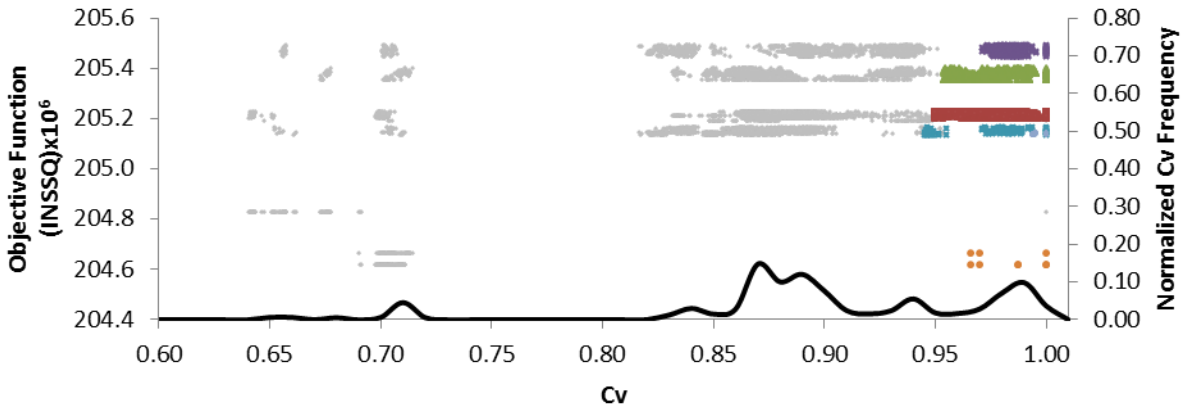


Figure 61: The INSSQ objective function versus C_v values for the 100 toroidal 6×6 rectangular SOMs trained by the MATLAB program with the REMD_hPTH4 data-subset (Table 13:8-qq). The scatter graph is of the INSSQ objective function. The C_v distribution (___) is along the bottom of the graph with the normalized values on the right vertical axis. The C_v is generated from two maps; for each map an objective function was computed (one C_v value has two objective functions). ◆ are all the C_v values, the coloured points are the two groups of maps which produce high C_v values with each other and have three or more maps in a group. ■ is the highest occupied group with 36 SOMs (group 1), ▲ has 28 SOMs (group 2), ✖ has 18 SOMs (group 3), ✸ has 10 SOMs (group 4), ✚ has 3 SOMs (group 5) ● has 3 SOMs (group 6).

8.3.5.2 Reproducible Clusters of hPTH in 10×10 Rectangular and Hexagonal SOMs

The 10×10 hexagonal SOMs generated using the MATLAB program for the REMD_hPTH4 data-subset (Table 13:8-ii) were partitioned into seven groups by the C_v comparison. Figure 62 shows the seven groups along with all other C_v values. The C_v distribution is along the bottom of the graph. The high C_v region is considered above 0.95. The largest three groups are considered reproducible. Group 2 (green) has a broad distribution of C_v values (~0.9-1.0). However, this is caused by a single SOM, if the SOM is removed from Group 2 all C_v values are above 0.95 and the group becomes compact. Only Groups 1, 2 and 3 are considered reproducible.

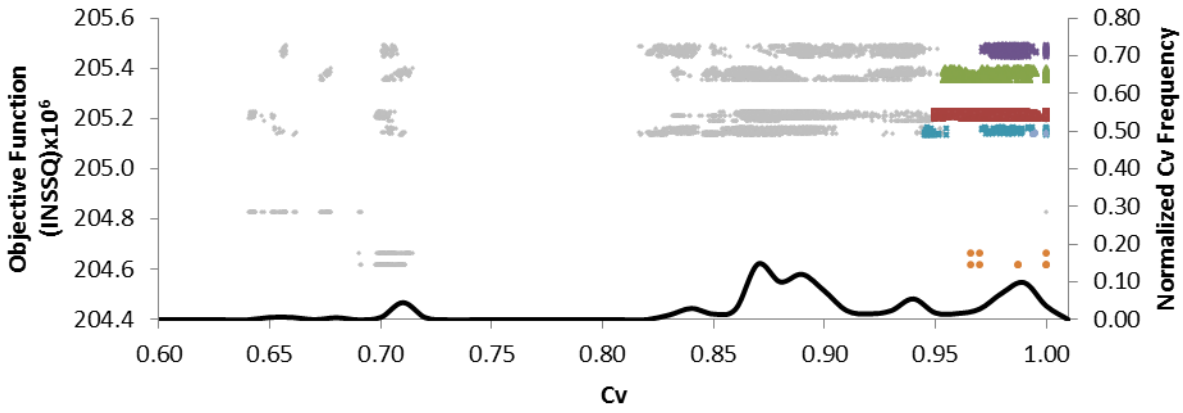


Figure 62: The INSSQ objective function versus C_v values of the 100 toroidal 10×10 hexagonal SOMs from the MATLAB program with the REMD_hPTH4 data-subset (Table 13:8-ii). The scatter graph is of the INSSQ objective function. The C_v distribution (—) is along the bottom of the graph with the normalized values on the right vertical axis. The C_v is generated from two maps; for each map an objective function was computed (one C_v value has two objective functions). ◆ are all the C_v values, the coloured points are the two groups of maps which produce high C_v values with each other and have three or more maps in a group. ■ is the highest occupied group with 27 SOMs (group 1), ▲ has 15 SOMs (group 2), and ✖ has 13 SOMs (group 3).

8.3.6 Comparison SOMs Trained by C++ and MATLAB Programs for the hPTH/REMD4 Data-Subset

In Section 7.3.9 it was shown for the conformational dataset describing MET, that SOMs can be reproducible and reproducibility is not dependent on the source code used. However, it is unclear how similar data partitions are from these two independent source codes. The C++ program produced two different partitionings of the REMD_hPTH4 data-subset (Section 8.3.4.4), while the MATLAB program produced one partitioning for each of the hexagonal and rectangular SOMs (Section 8.3.5.1). The four different partitionings of the hPTH/REMD4 conformational data-subset generated by different source code or nodal geometry were compared by their ${\chi^2}_{ij}$ values. Figure 63 shows the six ${\chi^2}_{ij}$ comparison between the four different partitionings of the SOMs produced by the different parameters (Table 13:8-mm, 8-pp, and 8-qq, where there are two partitionings for 8-mm). The dark grey sections are the values determined to be the high ${\chi^2}_{ij}$ values. The high ${\chi^2}_{ij}$ values were determined by values above 2000 or values

that plateau before a sudden drop in the χ^2_{ij} values. Therefore, the number of comparisons in the dark grey region changes for each comparison.

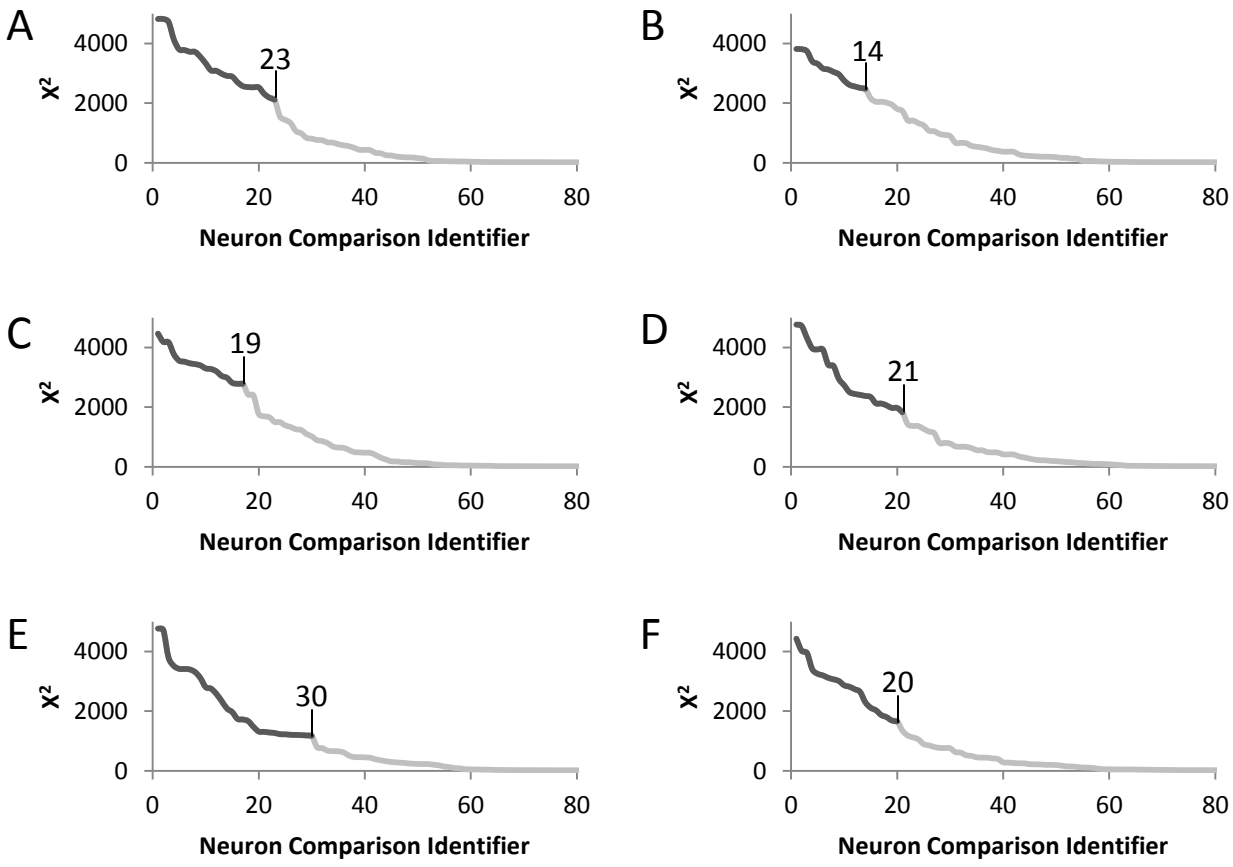


Figure 63: Comparison of SOMs produced by C++ and MATLAB programs of hPTH/REMD4 data-subset. Neuron Comparison Identifier is an assigned label. The top comparisons are highlighted in dark grey. (A) Group 1 vs Group 2 of the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm), (B) Group 1 of the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm) vs the toroidal 6×6 hexagonal SOMs from the MATLAB program (Table 13:8-pp), (C) Group 2 of the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm) vs the toroidal 6×6 hexagonal SOMs from the MATLAB program (Table 13:8-pp), (D) Group 1 of the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm) vs the toroidal 6×6 rectangular SOMs from the MATLAB program (Table 13:8-qq), (E) Group 2 of the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm) vs the toroidal 6×6 rectangular SOMs from the MATLAB program (Table 13:8-qq), and (F) the toroidal 6×6 hexagonal SOMs from the MATLAB program (Table 13:8-pp) vs the toroidal 6×6 rectangular SOMs from the MATLAB program (Table 13:8-qq).

Using the high χ^2_{ij} values (dark grey; Figure 63) the four SOMs were compared to each other. It was found that two nodes are identical between all the SOM partitionings (Figure 64). These two were the only nodes were determined to possess the same memberships for all four

6×6 SOMs. These two nodes contain identical nodal memberships for the two clusters coloured green and blue. The nodal centers of the green nodes have an root-mean-squared-deviation (RMDS) of 5.19 ± 1.03Å and the blue nodes have an RMS of 3.40 ± 2.21Å. The blue nodes by nodal centre conformation comparison are more similar to each other than the green nodes. The nodal tolerances of the green nodes are lower than the blue nodes. The conformations in these nodes were also found in adjacent nodes of the 10×10 hexagonal SOMs obtained by the MATLAB program (Table 13:8-ii). The members in the blue node were split between two nodes on the 10×10 SOM and the members of the green nodes were split among four nodes on the 10×10 SOM. When comparing the blue node from one map to the blue node on the other, the χ^2_{ij}values for a blue to blue comparison are always higher than the χ^2_{ij}value for a green to green comparison.

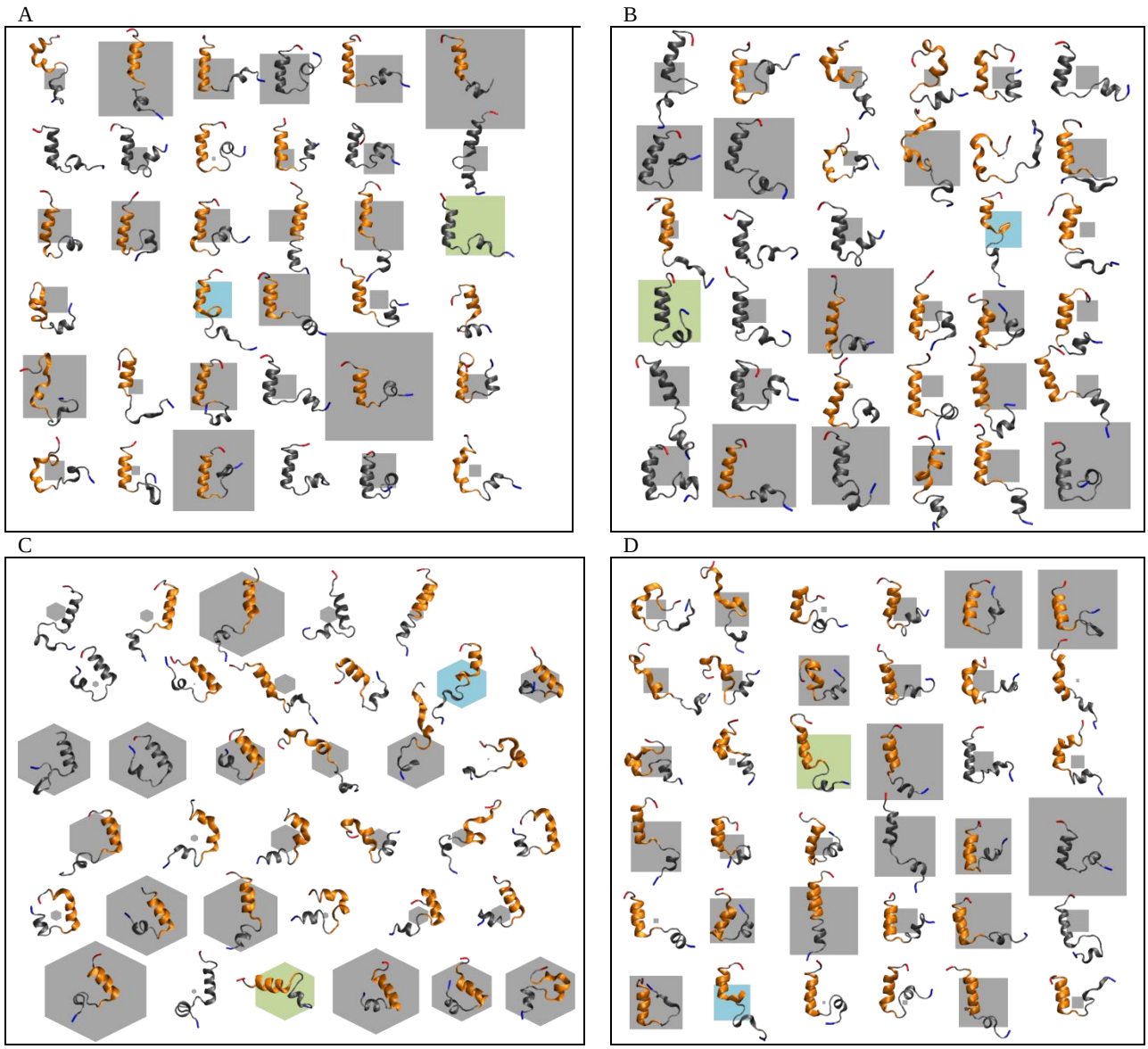


Figure 64: Examples of the four different partitions of the 6×6 SOMs with different geometry and programs. (A) Group 1 the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm), (B) Group 2 the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm), (C) the toroidal 6×6 hexagonal SOMs from the MATLAB program (Table 13:8-pp), and (D) the toroidal 6×6 rectangular SOMs from the MATLAB program (Table 13:8-qq). The two nodes that were found to be similar among all SOMs are coloured in green and blue.

Specific nodes were found to correlate between the all 6×6 SOMs (Figure 64) and 10×10 SOMs. However, the quantitative measurement of the similarity of the SOMs generated from different source codes as well as different geometries and sizes has not been fully explored. The C_v distributions were used to determine the similarity between different partitionings of the SOMs. The 10×10 hexagonal SOMs from the MATLAB program (Table 13:8-ii) are more

similar to the 6×6 SOMs than the 6×6 SOMs are to each other (Figure 65). Figure 65 shows a comparison of group 1 of the 6×6 SOM generated from the C++ program (Table 13:8-mm) compared to all other groups from the other programs. Group 1 was chosen as a starting partition. The 10×10 SOMs have higher C_v values when compared to Group 1 of the 6×6 SOM produced from the C++ Program, than any of the other 6×6 SOMs compared to Group 1. All three of the groups in the 10×10 hexagonal SOM overlap with each other (~0.75-0.86), while the 6×6 partitionings of the SOMs have more of a distribution (~0.66-0.77). From the C_v distribution it can be assumed group 1 of the 6×6 SOM generated from the C++ program (Table 13:8-mm) is more similar to all the 10×10 SOMs than the 6×6 SOMs.

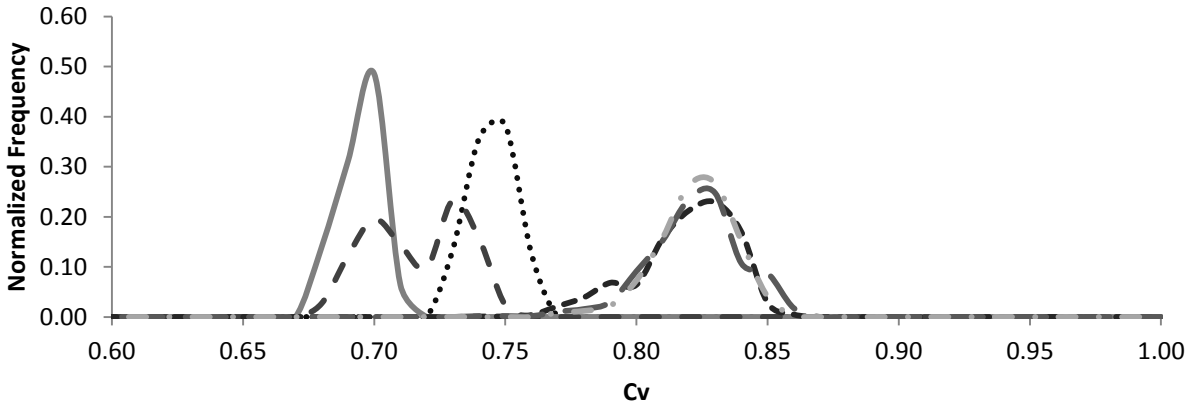


Figure 65: The three other partitionings of SOMs compared by C_v to Group 1 of the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm). (.....) Group 2 of the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm), (——) the toroidal 6×6 hexagonal SOMs from the MATLAB program (Table 13:8-pp), (– –) the toroidal 6×6 hexagonal SOMs from the MATLAB program (Table 13:8-pp), (– – –) Group 1 of the toroidal 10×10 hexagonal SOMs from the MATLAB program (Table 13:8-ii), (— —) Group 2 of the toroidal 10×10 hexagonal SOMs from the MATLAB program (Table 13:8-ii), and (— —) Group 3 of the toroidal 10×10 hexagonal SOMs from the MATLAB program (Table 13:8-ii).

Knowing the 10×10 SOMs from the MATLAB program (Table 13:8-ii) are more similar to the 6×6 SOMs than the 6×6 SOMs are to each other (Figure 65), we explored which of the 10×10 SOM groups are more similar than the others. It was found by the C_v distributions that group 3 of the 10×10 hexagonal SOMs from the MATLAB program (Table 13:8-ii) had the most

similar partitioning to all of the 6×6 SOMs than the other groups generated from the 10×10 maps. Group 3 was found 13 times from the 100 independently trained SOMs. Figure 66 shows the C_v distributions for each of the main four partitionings of the 6×6 SOMs compared to group 3 of the 10×10 SOMs. All of the C_v distributions overlap in a narrow range between 0.75 and 0.85. This is a higher range than comparing the 6×6 SOMs to each other (0.62-0.78). The other groups had a broader distribution between 0.70 and 0.85. For the hPTH/REMD4 conformational data-subset, 100 independently trained SOMs were enough to locate different data partitionings reproducibly. For the conformational data-sub set hPTH/REMD4, the 10×10 SOMs were better at clustering the conformational data than the 6×6 SOMs because all of the different parttionings could be found with high C_v values for the group 3 of the 10×10 hexagonal SOMs from the MATLAB program (Table 13:8-ii).

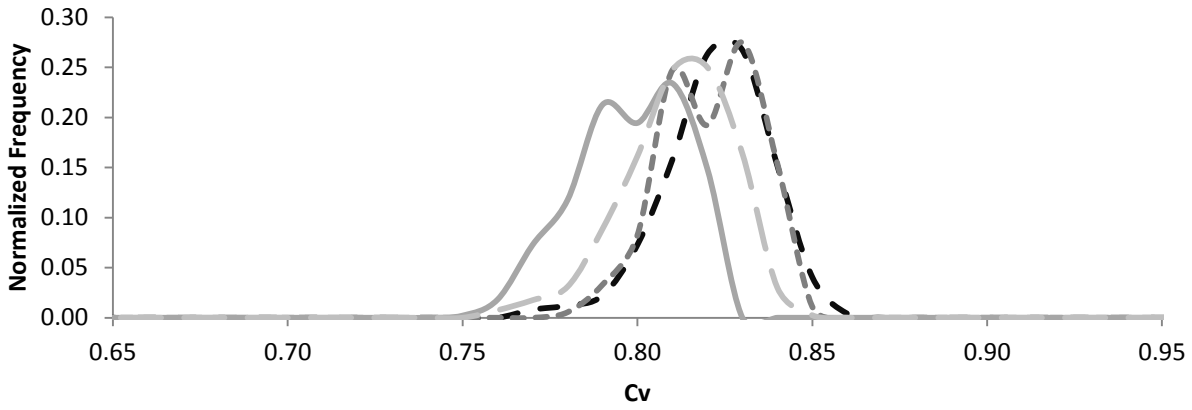


Figure 66: The four different SOMs compared by C_v to Group 3 of the toroidal 10×10 hexagonal SOMs from the MATLAB program (Table 13:8-ii). (— —) Group 1 the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm), (– – –) Group 2 the toroidal 6×6 rectangular SOMs from the C++ program (Table 13:8-mm), (— —) the toroidal 6×6 hexagonal SOMs from the MATLAB program (Table 13:8-pp), and (——) the toroidal 6×6 rectangular SOMs from the MATLAB program (Table 13:8-qq).

8.4 Conclusion of hPTH

The use of dPCA to reduce the number of variables describing protein conformations makes it easier to find relevant partitionings of the data. Even though the protein hPTH is much

larger (34-amino acids) than the peptide MET (5-amino acids), the SOMs could still find reproducible clusters using only 100 independently produced SOMs. However, in order for this to be true, the SOM must be optimized for training length. If the SOMs are not trained for a sufficient number of iterations, there will be a limited number of reproducible maps (i.e. low C_v values).

Comparing experimental structures of hPTH bound to its receptor to conformations produced from simulations of the free hPTH, can be used to suggest whether or not the protein changes its conformation upon binding. The X-ray structures of the C-terminal region of hPTH bound to the PTHR can be classified by supervised training using SOMs produced from unbound hPTH REMD conformations. Therefore, the first stage of hPTH binding does not induce a conformational change of the C-terminus. When a 6×6 rectangular SOM from hPTH/REMD4 was used for the supervised classification of the conformations from the classical MD simulations, it was seen that in most cases MD simulations explore a subset of REMD simulations. We found the same results as Chu *et al.*[97] and Gordon and Somorjai,[2] that MD simulations do not explore the breadth of conformational space that hPTH can explore via REMD simulations.

Using $\chi^2{}_{ij}$ values to compare partitionings of data from different SOM programs (C++ and MATLAB) and different geometries (rectangular and hexagonal), determined that some conformations stay together on the map. MATLAB outperformed the C++ program when the SOM mapsize was increased to a 10×10 map. However, when the mapsize is 6×6, than the no one program could not be ranked as better than the other. Group 3 of the toroidal 10×10 hexagonal SOMs from the MATLAB program could find the same partitionings within all the 6×6 SOMs from both programs (C++ and MATLAB), as well as for those having different

geometries (rectangular and hexagonal). The 10×10 SOMs outperformed the 6×6 SOMs for partitioning the conformations of hPTH. Further exploration must be done on the geometries of the SOMs, to determine if rectangular is better than hexagonal or vice versa. The revised similarity index, rSI, equation [25], could be used to help determine which geometry is better, since rSI includes a description of neighbourhoods in trained SOMs.

9 Future Work

During this work, we explored aspects of SOMs for clustering conformations obtained by MD simulations. We determined:

- SOMs are reproducible when run multiple times (≤100)
- SOMs need an evaluation which compared nodal membership (C_v , rSI) to group data partitionings
- Toroidal SOMs cluster data better than bordered SOMs
- The training length must be optimized for each system
- Decreasing neighbourhood function cluster the data better than constant neighbourhood function
- SOM parameterization was done with both C_v and an objective function (LINEAR, 0.05, sequential training algorithm)

The SOM analysis still needs to be explored for:

- rSI should be explored; since it has a neighbourhood component it could give insight as to which nodal geometry and/or nodal neighbourhood is better
- Develop a criteria to determine if the mapsize is appropriate for the system being studied
- Exploration of nodal geometry and nodal neighborhoods should help differentiate between the different source codes
- Comparing different partitionings by nodal conformational membership, to determine how similar or different the partitionings are
- Explore a system with more experimental results to explore supervised training of SOMs; Nuclear Co-activation Binding Domain (NCBD) is a protein which is mostly unstructured until bound to one of its ligands. There are free (PDB: 2KKJ,[99] 1JJS[100]) and bound (PDB: 1KBH,[101] 1ZOQ,[102] 2C52[103]) NMR structures in the PDB.

10 Conclusion

The SOMs have the ability of finding reproducible clusters between different source codes, mapsizes, as well as map geometries. The SOMs also have the ability to determine how similar a dataset is to another, though the methodology is slightly different (supervised training

of an SOM). The SOMs must be run multiple times to produce reproducible clusters. The maps must be evaluated by a form of nodal or neighbourhood tightness as well as a nodal membership comparison to elucidate the different groups of partitions in multiple SOM runs. Through this work C_v comparisons were used, since the use of C_v index gave the same result as the rSI comparison and was a faster computation; the C_v comparisons had the added benefit of relating nodes 1:1 through ${\chi^2}_{ij}$ values. Since the rSI includes a neighbourhood components while the C_v dose not, rSI might be a better comparison for different geometries.

References

(1) Duda, R.; Hart, P. E.; Stork, D. *Pattern Classification*; John Wiley & Sons: New York, 2001.

(2) Gordon, H. L.; Somorjai, R. L. *Proteins* **1992**, *14* (2), 249.

(3) Kohonen, T. *Neural Networks* **2006**, *19*, 723.

(4) Hyvönen, M. T.; Hiltunen, Y.; El-Deredy, W.; Ojala, T.; Vaara, J.; Kovanen, P. T.; Ala-Korpela, M. *J. Am. Chem. Soc.* **2001**, *123* (15), 810.

(5) Shenkin, P. S.; McDonald, D. Q. *J. Comput. Chem.* **1994**, *15* (8), 899.

(6) Ding, C.; He, X. *Twentyfirst Int. Conf. Mach. Learn. ICML 04* **2004**, *Cl*, 29.

(7) Wright, P. E.; Dyson, H. J. *J. Mol. Biol.* **1999**, *293*, 321.

(8) Hartl, F. U.; Hayer-Hartl, M. *Nat. Struct. Mol. Biol.* **2009**, *16* (6), 574.

(9) Education, N. Scitable http://www.nature.com/scitable/ebooks/essentials-of-cell-biology-14749010/how-do-cells-decode-genetic-information-into-14751777 (accessed Aug 21, 2014).

(10) Altis, A.; Nguyen, P. H.; Hegger, R.; Stock, G. *J. Chem. Phys.* **2007**, *126*.

(11) Edison, S. *Nat. Struct. Biol.* **2001**, *8* (3), 201.

(12) Ramachandran, G. N.; Ramakrishnan, C.; Sasisekharan, V. *J. Mol. Biol.* **1963**, *7* (1), 95.

(13) Voet, D.; Voet, J. In *Biochemistry*; John Wiley & Sons, 2004; pp 222–223.

(14) The Ramamchandran Plot http://www.iop.vast.ac.vn/theor/conferences/smp/1st/kaminuma/UCSFComputerGraphicsLab/AAA.html.

(15) Andrews, F. In *Equilibrium Statistical Mechanics*; John Wiley & Sons: New York, 1975; pp 183–186.

(16) Fraccalvieri, D.; Tiberti, M.; Pandini, A.; Bonati, L.; Papaleo, E. *Mol. Biosyst.* **2012**, *8*, 2680.

(17) Shao, J.; Tanner, S. W.; Thompson, N.; Cheatham, T. E. *J. Chem. Theory Comput.* **2007**, *3*, 2312.

(18) Rajabpour, A.; Akizi, F.; Heyhat, M.; Gordiz, K. *Int. Nano Lett.* **2013**, *3* (1), 58.

(19) Leach, A. R. In *Molecular Modelling: Principles and Applications*; Longman, 1996; pp 303–315.

(20) Lisboa, P. J. G.; Etchells, T.; Jarman, I. H.; Chambers, S. J. *BMC Bioinformatics* **2013**, *14 Suppl 1* (Suppl 1), S8.

(21) Kohonen, T. *Biol. Cybern.* **1982**, *43*, 59.

(22) Kasson, P. M.; Kelley, N. W.; Singhal, N.; Vrljic, M.; Brunger, A. T.; Pande, V. S. *Proc. Natl. Acad. Sci. U. S. A.* **2006**, *103* (32), 11916.

(23) Kohonen, T. *Proceedings of the IEEE*. 1990, pp 1464–1480.

(24) Kirew, D. B.; Chretien, J. R.; Bernard, P.; Ros, F. *SAR QSAR Environ. Res.* **1998**, *8* (i), 93.

(25) Zaït, M.; Messatfa, H. *Futur. Gener. Comput. Syst.* **1997**, *13* (97), 149.

(26) López-Rubio, E.; DíazRamos, A. *Neural Networks* **2014**, *56*, 35.

(27) Ballabio, D.; Vasighi, M.; Filzmoser, P. *Anal. Chim. Acta* **2013**, *765*, 45.

(28) Allen, M. P.; Tildesley., D. J. In *Computer simulation of liquids*; Clarendon Press: New York, 1989; p 30.

(29) Valova, I.; Beaton, D.; Buer, A.; MacLean, D. *Neural Comput. Appl.* **2010**, *19*, 953.

(30) Kiang, M. Y.; Kulkarni, U. R.; St Louis, R. *J. Oper. Res. Soc.* **2001**, *52* (1), 93.

(31) Vesanto, J.; Himberg, J.; Alhoniemi, E.; Parhankangas, J. *Tech. Rep. A57* **2000**, *2* (0), 59.

(32) Merkow, M.; DeLisle, R. K. *J. Chem. Inf. Model.* **2007**, *47*, 1797.

(33) Garge, N. R.; Page, G. P.; Sprague, A. P.; Gorman, B. S.; Allison, D. B. *BMC Bioinformatics* **2005**, *6 Suppl 2* (2), S10.

(34) Cramer, H. *Mathematical Methods of Statistics*; Princeton Univercity Press: Princeton, NJ, 1946.

(35) Chen, P.; Popovich, P. *Correlation*; Lewis-Beck, M., Ed.; SAGE Publications, Inc.: 2455 Teller Road, Thousand Oaks California 91320 United States of America, 2002.

(36) Vesanto, J.; Himberg, J.; Alhoniemi, E.; Parhankangas, J. *Proc. Matlab DSP Conf.* **1999**, *99*, 16.

(37) Murtola, T.; Kupiainen, M.; Falck, E.; Vattulainen, I. *J. Chem. Phys.* **2007**, *126* (May 2014).

(38) Fraccalvieri, D.; Pandini, A.; Stella, F.; Bonati, L. *BMC Bioinformatics* **2011**, *12* (1), 158.

(39) Bouvier, G.; Duclert-Savatier, N.; Desdouits, N.; Meziane-Cherif, D.; Blondel, A.; Courvalin, P.; Nilges, M.; Malliavin, T. E. *J. Chem. Inf. Model.* **2014**, *54* (Figure 2), 289.

(40) Extended Kohonen Maps http://www.let.rug.nl/kleiweg/kohonen/.

(41) Kleiweg, P. Neurale netwerken: Een inleidende cursus met practica voor de studie Alfa-Informatica., 1996.

(42) Erwin, E.; Obermayer, K.; Schulten, K. *Biol. Cybern.* **1992**, *67* (1990), 47.

(43) Neme, A.; Chavez, E.; Cervera, A.; Mireles, V. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* **2008**, *5163 LNCS* (PART 1),

671.

(44) González, M. *École thématique la Société Française la Neutron.* **2011**, *12*, 169.

(45) Rappe, A. K.; Casewit, C. J.; Colwell, K. S.; Goddard, W. A.; Skiff, W. M. *J. Am. Chem. Soc.* **1992**, *114* (25), 10024.

(46) Steffen, C.; Thomas, K.; Huniar, U.; Hellweg, A.; Rubner, O.; Schroer, A. *J. Comput. Chem.* **2010**, *31*, 2967.

(47) Brooks, B. R.; Bruccoleri, R. E.; Olafson, B. D.; States, D. J.; Swaminathan, S.; Karplus, M. *J. Comput. Chem.* **1983**, *4*, 187.

(48) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. *J. Comput. Chem.* **2005**, *26*, 1781.

(49) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. *J. Chem. Phys.* **1983**, *79*, 926.

(50) Dill, K.; Bromberg, S. In *Molecular Driving Forces: Statistical Thermodynamics in Biology, Chemistry, Physics, and Nanoscience*; 2010; pp 45–51.

(51) Ryckaert, J.-P.; Ciccotti, G.; Berendsen, H. J. . *J. Comput. Phys.* **1977**, *23*, 327.

(52) Alder, B. J.; Wainwright, T. E. *J. Chem. Phys.* **1959**, *31*, 459.

(53) Darden, T.; York, D.; Pedersen, L. *J. Chem. Phys.* **1993**, *98*, 10089.

(54) Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pedersen, L. G. *J. Chem. Phys.* **1995**, *103* (19), 8577.

(55) Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; DiNola, A.; Haak, J. R. *J. Chem. Phys.* **1984**, *81* (8), 3684.

(56) Swendsen, R. H.; Wang, J. S. *Phys Rev Lett* **1986**, *57* (21), 2607.

(57) Zhou, R. *Methods Mol. Biol.* **2007**, *350* (November), 205.

(58) Earl, D. J.; Deem, M. W. *Phys. Chem. Chem. Phys.* **2005**, *7* (23), 3910.

(59) Sanbonmatsu, K. Y.; García, A. E. *Proteins Struct. Funct. Genet.* **2002**, *46* (2), 225.

(60) Periole, X.; Mark, A. E. *J. Chem. Phys.* **2007**, *126*.

(61) Woods, C. J.; Essex, J. W.; King, M. A. *J. Phys. Chem. B* **2003**, *107* (49), 13703.

(62) Daffertshofer, A.; Lamoth, C. J. C.; Meijer, O. G.; Beek, P. J. *Clin. Biomech.* **2004**, *19*, 415.

(63) Christie, O. H. J. *Chemom. Intell. Lab. Syst.* **1995**, *29* (95), 177.

(64) Fischer, S. L.; Hampton, R. H.; Albert, W. J. *Computer Methods in Biomechanics and Biomedical Engineering*. Taylor & Francis 2012, pp 1–5.

(65) Glykos, N. M. *J. Comput. Chem.* **2006**, *27* (14), 1765.

(66) Glykos, N. M. Structural and Computational Biology http://utopia.duth.gr/~glykos/carma.html (accessed Sep 3, 2013).

(67) Mu, Y.; Nguyen, P. H.; Stock, G. *PROTEINS Struct. Funct. Gen.* **2005**, *58* (May 2004), 45.

(68) Garbay-Jaureguiberry, C.; Roques, B. P.; Oberlin, R. *Biochem. Biophys. Res. Commun.* **1976**, *71* (2), 558.

(69) Hughes, J.; Smith, T. W.; Kosterlitz, H. W.; Fothergill, L.; Morgan, B.; Morris, H. R. *Nature* **1975**, *258* (5536), 577.

(70) Spirtes, M. A.; Schwartz, R. W.; Mattice, W. L.; Coy, D. H. *Biochem. Biophys. Res. Commun.* **1978**, *81* (2), 602.

(71) Marcotte, I.; Separovic, F.; Auger, M.; Gagné, S. M. *Biophys. J.* **2004**, *86* (March), 1587.

(72) Perez, J. J.; Villar, H. O.; Loew, G. H. *J. Comput. Aided. Mol. Des.* **1992**, *6*, 175.

(73) Zhan, L.; Chen, J. Z. Y.; Liu, W.-K. *Biophys. J.* **2006**, *91* (7), 2399.

(74) Malevanets, A.; Wodak, S. J. *Biophys. J.* **2011**, *101* (4), 951.

(75) Zaman, M. H.; Shen, M. Y.; Berry, R. S.; Freed, K. F. *J. Phys. Chem. B* **2003**, *107*, 1685.

(76) Westbrook, J.; Feng, Z.; Chen, L.; Yang, H.; Berman, H. M. *Nucleic Acids Res.* **2003**, *31* (1), 489.

(77) Humphrey, W.; Dalke, A.; Schulten, K. *J. Mol. Graph.* **1996**, *14* (October 1995), 33.

(78) In *This work was made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET:http://www.sharcnet.ca) and Compute/Calcul Canada*.

(79) Gienow, M. C. Conformational Clustering of Peptide Met-enkenphalin Employing a Self-Organizing Map with Toroidal Boundaries, Brock University, 2013.

(80) Mount, N. J.; Weaver, D. *Pattern Anal. Appl.* **2011**, *14*, 139.

(81) Weaver, R. E.; Wigglesworth, M. J.; Donnelly, D. *Peptides* **2014**, *61*, 83.

(82) Caporale, A.; Woznica, I.; Schievano, E.; Mammi, S.; Peggion, E. *Amino Acids* **2010**, *38* (4), 1269.

(83) Jin, L.; Briggs, S. L.; Chandrasekhar, S.; Chirgadze, N. Y.; Clawson, D. K.; Schevitz, R. W.; Smiley, D. L.; Tashjian, A. H.; Zhang, F. *J. Biol. Chem.* **2000**, *275* (35), 27238.

(84) Vilardaga, J. P.; Gardella, T. J.; Wehbi, V. L.; Feinstein, T. N. *Trends Pharmacol. Sci.* **2012**, *33* (8), 423.

(85) Stroup, J.; Kane, M. P.; Abu-Baker, A. M. *Am. J. Heal. Pharm.* **2008**, *65* (6), 532.

(86) Castro, M.; Nikolaev, V. O.; Palm, D.; Lohse, M. J.; Vilardaga, J.-P. *Proc. Natl. Acad. Sci. U. S. A.* **2005**, *102* (44), 16084.

(87) Hoare, S. R. J.; Gardella, T. J.; Usdin, T. B. *J. Biol. Chem.* **2001**, *276* (11), 7741.

(88) Gardella, T. J.; Jüppner, H. *Trends Endocrinol. Metab.* **2001**, *12* (5), 210.

(89) Potetinova, Z.; Barbier, J.-R.; Suen, T.; Dean, T.; Gardella, T. J.; Willick, G. E. *Biochemistry* **2006**, *45* (37), 11113.

(90) Schievano, E.; Mammi, S.; Silvestri, L.; Behar, V.; Rosenblatt, M.; Chorev, M.; Peggion, E. *Biopolymers* **2000**, *54*, 429.

(91) Thomas, B. E.; Woznica, I.; Mierke, D. F.; Wittelsberger, A.; Rosenblatt, M. *Mol. Endocrinol.* **2008**, *22* (5), 1154.

(92) Pioszak, A.; Xu, H. E. *Proc. Natl. Acad. Sci.* **2008**, *105* (13), 5034.

(93) Caporale, A.; Gesiot, L.; Sturlese, M.; Wittelsberger, A.; Mammi, S.; Peggion, E. *Amino Acids* **2012**, *43* (1), 207.

(94) Marx, U. C.; Adermann, K.; Bayer, P.; Forssmann, W. G.; Rösch, P. *Biochem. Biophys. Res. Commun.* **2000**, *267*, 213.

(95) Marx, U. C.; Austermann, S.; Bayer, P.; Adermann, K.; Ejchart, A.; Sticht, H.; Walter, S.; Schmid, F.-X.; Jaenicke, R.; Forssmann, W.-G.; Rosch, P. *J. Biol. Chem.* **1995**, *270* (25), 15194.

(96) Pellegrini, M.; Royo, M.; Rosenblatt, M.; Chorev, M.; Mierke, D. F. *J. Biol. Chem.* **1998**, *273* (17), 10420.

(97) Chu, J. W.; Yin, J.; Wang, D. I. C.; Trout, B. L. *Biochemistry* **2004**, *43*, 14139.

(98) Roux, B.; Simonson, T. *Biophys. Chem.* **1999**, *78* (1–2), 1.

(99) Kjaergaard, M.; Teilum, K.; Poulsen, F. M. *Proc. Natl. Acad. Sci. U. S. A.* **2010**, *107* (1), 12535.

(100) Lin, C. H.; Hare, B. J.; Wagner, G.; Harrison, S. C.; Maniatis, T.; Fraenkel, E. *Mol. Cell*

2001, *8*, 581.

(101) Demarest, S. J.; Martinez-Yamout, M.; Chung, J.; Chen, H.; Xu, W.; Dyson, H. J.; Evans, R. M.; Wright, P. E. *Nature* **2002**, *415* (January), 549.

(102) Qin, B. Y.; Liu, C.; Srinath, H.; Lam, S. S.; Correia, J. J.; Derynck, R.; Lin, K. *Structure* **2005**, *13*, 1269.

(103) Waters, L.; Yue, B.; Veverka, V.; Renshaw, P.; Bramham, J.; Matsuda, S.; Frenkiel, T.; Kelly, G.; Muskett, F.; Carr, M.; Heery, D. M. *J. Biol. Chem.* **2006**, *281*, 14787.